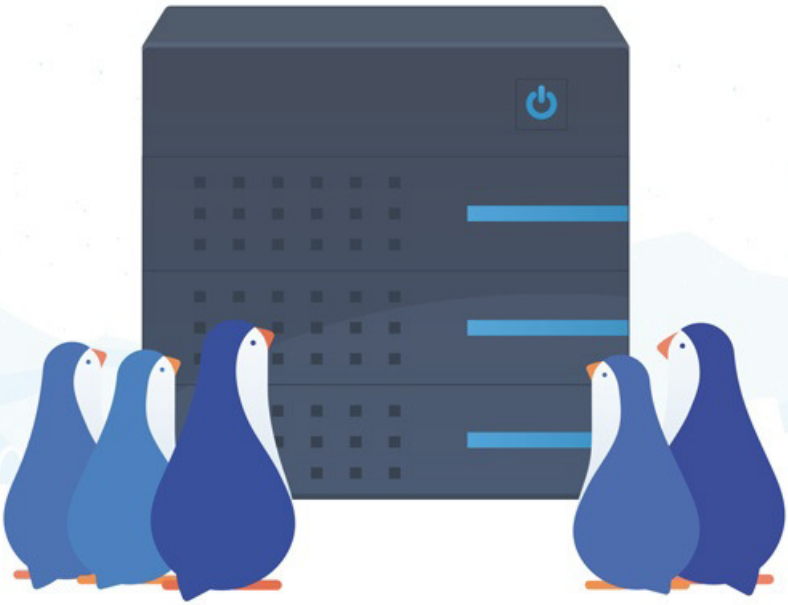
Guia prático do Servidor Linux

Administração Linux para iniciantes



Casa do Código

JULIANO RAMOS

ISBN

Impresso e PDF: 978-85-94188-78-6

EPUB: 978-85-94188-79-3

MOBI: 978-85-94188-80-9

Caso você deseje submeter alguma errata ou sugestão, acesse http://erratas.casadocodigo.com.br.

PREFÁCIO

O mundo da tecnologia pode ser muito contraditório. No cerne da iniciativa de desenvolver qualquer produto está sempre a intenção de que este produto chegue até o público final, e quando falamos de software o chamado “público final” ganha o nome de “usuário".

Curiosamente, na cultura dos softwares, distribuir conhecimento, além do software em si, não é algo profundamente cultivado. Felizmente existe a cultura de desenvolvimento *open source* que fomenta este lado de contribuição e colaboração das pessoas em outros segmentos da sociedade também.

Atualmente eu sou autor de um dos maiores blogs sobre o assunto “Linux” em língua portuguesa e tenho contato diário com pessoas de todos os tipos, de todos os sonhos. Há alguns anos eu tive a oportunidade de conhecer alguém que tem um espírito de compartilhamento de conhecimento que converge muito com os meus esforços.

Como a maior parte conhece, o professor Juliano Ramos é um exemplo de como compartilhar conhecimento pode trazer coisas boas para nossas vidas. Se para muitos ele é “o Professor Juliano Ramos”, eu simplesmente o conheci como “Juliano”, uma pessoa de mente inquieta e pesquisadora, esforçada e detalhista, alguém que sacrificou horas de seus dias em prol de projetos maiores e de ajudar as pessoas. Neste exato momento, você está lendo parte desse esforço.

Dizem que quando você se apaixona por algo e torna isso o seu

trabalho e lazer ao mesmo tempo só podem sair coisas boas, além da própria satisfação em fazer parte de um projeto que nós mesmos criamos, que extrapolam folhas escritas e páginas de internet. É exatamente este tipo de esforço que você encontra nesta obra.

Um pré-requisito para ensinar é aprender antes (e muitas vezes durante) a nossa jornada como profissionais. O professor Juliano Ramos é gabaritado e reconhecido através de certificações como a *LPCI-2*, *SUSE Certified Linux Administrator* (CLA) e *Red Hat*, mas além disso, ele tem outra coisa que é muito importante na hora de passar conhecimento, experiência de campo.

Trabalhando com Linux há mais de 12 anos já perdeu a conta de quantos alunos devem ter passados por suas mãos, e eu, como professor que sou e fui (durante 7 anos) sei o quanto é fácil perder as contas e ao mesmo tempo lembrar dos sorrisos dos alunos queridos que conseguiram atingir seus objetivos profissionais e de vida através dos nossos ensinamentos.

Uma das grandes chaves de se ensinar é ter a habilidade de voltar no tempo e preparar uma aula para um aluno completamente novo, assim como nós já fomos um dia. Este livro tem este exato enfoque. Se você for novo no mundo Linux não se preocupe, o conteúdo é muito simples e mastigado para que qualquer pessoa que tenha tido um mínimo de contato possa avançar sem problemas e obter conhecimento, um passo de cada vez.

Cada passo que você der vai deixá-lo mais próximo de se tornar um profissional completo e mais experiência. O livro é ao mesmo tempo muito prático, então não tenha medo de montar o

seu próprio ambiente para testar tudo o que for proposto por aqui. Teoria e prática andam juntas nessa obra.

Depois de absorver tudo o que for possível deste livro, que você sinta o mesmo sentimento que o Juliano e eu compartilhamos, a sede de aprender e ensinar. Transmita o conhecimento, a cultura *open source* não precisa ficar só no código, faça com que a sociedade possa crescer graças às suas contribuições intelectuais também.

Não vamos nos alongar mais, certo? Aproveite a leitura e eu o vejo na internet afora através do blog e do canal Diolinux.

Dionatan Simioni,

https://www.diolinux.com.br https://youtube.com/Diolinux

FORMAÇÃO SYSADMIN LINUX

Este livro tem como objetivo formar profissionais na Administração de servidores GNU/Linux.

Agradecimentos

Agradeço a Deus Jeová, pelas conquistas diárias, pela pequena força que me concede de ver o sol brilhar, de poder aprender coisas novas todos os dias, e de, nas noites por mais frias e tenebrosas, encontrar nele o conforto necessário para saber que o sol brilhará de novo no outro dia. Agradeço ao meu eterno e único amor, Fernanda Santos, que sempre me incentivou a escrever um livro sobre o Linux. Agradeço à minha filha Giovanna, que me permitiu por diversos momentos trabalhar, em vez de jogar ou brincar. Minha eterna Gigi, carrego você em meus pensamentos, coração e na minha tatuagem nova é claro, ao lado do pinguim do Tux. Para finalizar, agradeço à Vivian Matsui, editora da Casa do Código, pela paciência e compreensão, deixando-me livre para escrever este livro no meu tempo. - Vamos que vamos!

Sobre o autor

Juliano Ramos é certificado LPCI-2 - Linux Professional Institute Certificate, SUSE Certified Linux Administrator (CLA) e Red Hat. Trabalha exclusivamente com Linux desde 2005. Professor Juliano, como gosta de ser chamado, já perdeu a conta de quantos alunos formou em seus cursos on-line e em escolas presenciais que já ministrou treinamento. Somente em cursos on-line seguramente já formou mais de 3.000 alunos em

administração de sistema Linux. Atualmente, o professor mantém sua própria empresa de cursos on-line de formação Linux chamada: Certificações NET.BR - https://www.certificacoes.net.br

Público-alvo

Este livro foi escrito para pessoas que querem se tornar um administrador de servidores Linux e desejam se profissionalizar e/ou conhecer mais sobre sua arquitetura e seus principais servidores. Apesar de ser um livro técnico, na medida do possível, procuro escrever com uma linguagem de simples compreensão, com base nas dúvidas comuns dos meus alunos e de pessoas que me seguem em meu canal no YouTube (https://youtube.com/profjulianoramos).

Sumário

1 Introdução **1**

1.1 Preparando o ambiente de estudo 2

1.2 Introdução teórica 3

1.3 Conclusão 5

2 Primeiros passos **6**

2.1 Terminal virtual 6

2.2 Logon 7

2.3 Iniciando pelo shell 8

2.4 Configurando o teclado no Debian 10

2.5 Histórico de comandos 11

2.6 O comando fc 12

2.7 Logout 13

2.8 Desligando o computador 13

2.9 Reiniciando o computador 14

2.10 Acessando diretórios 15

2.11 Acessando a raiz 16

2.12 Teclas de atalhos 16

2.13 Conclusão 17

3 Obtendo ajuda **18**

3.1 A busca pelo conhecimento 18

3.2 As man pages 19

3.3 Formas de documentação 19

3.4 Comandos de ajuda 20

3.5 Conclusão 24

4 Comandos GNU/Linux **25**

4.1 O comando ls 25

4.2 ls -a 28

4.3 ls -R 28

4.4 Criar arquivo 28

4.5 Curingas 28

4.6 Criando diretórios 30

4.7 Removendo arquivos/diretórios 30

4.8 Copiar arquivos/diretórios 31

4.9 Movendo arquivos 31

4.10 Conclusão 32

5 FHS - Hierarquia de arquivos **33**

5.1 Estrutura de diretórios 33

5.2 Conclusão 39

6 Editor de texto VIM **40**

6.1 Editor de texto VIM 40

6.2 Como interpretar atalhos e comandos 41

6.3 Os modos do VIM 42

6.4 Abrindo caminhos dentro de um arquivo 42

6.5 Conclusão 48

7 Primeiros passos no shell script 49

7.1 Declarando e usando variáveis 53

7.2 Comando de seleção ou tomada de decisão 55

7.3 Loops condicionais 60

7.4 Funções 61

7.5 Conclusão 63

8 Introdução a redes 64

8.1 Os protocolos TCP/IP 64

8.2 Configuração dinâmica 70

8.3 Configuração estática 72

8.4 Configurando hostname 73

8.5 Conclusão 75

9 Instalação, remoção e atualização de programas 76

9.1 Gerenciando pacotes no Debian 77

9.2 Conclusão 87

10 Servidor SSH 89

10.1 SSH com chaves assimétricas 94

10.2 Conclusão 95

11 Particionamento de disco 97

11.1 Tipos de partições 98

11.2 O particionador FDISK 100

11.3 O particionador CFDISK 102

11.4 Aplicando um sistema de arquivos 104

11.5 Memória SWAP 107

11.6 Conclusão 108

12 Quotas de disco 109

12.1 Conclusão 113

13 Arquitetura do kernel Linux 114

13.1 Trabalhando com módulos 115

13.2 Compilando um kernel 120

13.3 Conclusão 123

14 Hardening 124

14.1 Conclusão 131

15 Servidor NFS - Compartilhando arquivos 133

15.1 Conclusão 135

16 Servidor RAID 137

16.1 Criando nosso laboratório 140

16.2 Conclusão 144

17 Logical Volume Manager - LVM3 145

17.1 Conclusão 148

18 Servidor SAMBA - Controlador de domínio 149

18.1 Conclusão 155

19 Servidor Apache 156

19.1 Instalando o servidor Apache 156

19.2 Instalação de um banco de dados 158

19.3 Conclusão 160

20 Servidor Proxy 161

20.1 Conclusão 169

Capítulo 1

INTRODUÇÃO

O Linux é o sistema operacional mais seguro que você poderá ter em mãos. Ele é amado por hackers, que estudam vulnerabilidades e falhas do sistema o tempo todo e compartilham soluções e correções. Empresas como a Dell, Asus e Acer produzem regularmente computadores que utilizam o Linux. Já grandes empresas, como IBM e Google, utilizam-no como estratégia em seus ambientes corporativos. Hoje, praticamente toda a infraestrutura da internet atua sobre o sistema do pinguim. Com o avanço da tecnologia, o Linux embarcado, em eletrônicos, eletrodomésticos e até carros, será cada vez mais comum. Conhecer e utilizar o Linux para qualquer pessoa que pretende se tornar um profissional em TI não é uma questão opcional, é um pré-requisito.

Tipografia

Quando exemplificarmos um comando iniciado com `#` (sustenido). Significa que este comando deve ser executado como *root*, que é o usuário administrador do Linux. Exemplo:

```
# apt-get upgrade
```

Quando o comando iniciar com `$` (cifrão), significa que é um comando que pode ser executado com qualquer usuário do seu

sistema. Exemplo:

```
$ apt-cache search vim
```

Para teclas que devem ser pressionadas, utilizarei tags < > , exemplo:

```
< CTRL > + < d >
```

1.1 PREPARANDO O AMBIENTE DE ESTUDO

Simular uma infraestrutura de rede de computadores hoje em dia é muito fácil e com um custo zero, graças ao poder da virtualização. Através da virtualização, podemos instalar e utilizar o Linux, indiferentemente do sistema nativo que esteja instalado fisicamente em nosso computador.

Virtualbox

Todo o nosso laboratório de estudo será com o uso do software de virtualização Virtualbox. Além de ser um software de código aberto, ele é gratuito e funciona em praticamente todos os sistemas operacionais da atualidade.

Download e instalação

No Debian, Ubuntu e distribuições derivadas, o processo de instalação é quase automatizado, já que este software está disponível nos repositórios de softwares destes sistemas. Você pode procurar pelo Virtualbox na sua central de programas, ou executar o comando a seguir em algum terminal:

```
# apt-get install virtualbox
```

Para instalar no Windows, Mac OS ou em outras distribuições de Linux, recomendo que acesse a página oficial do Virtualbox: https://www.virtualbox.org/wiki/Downloads

Sistemas operacionais

Após a instalação do Virtualbox, faça a instalação dos seguintes sistemas operacionais:

- **Bunsenlabs** - Esta é uma customização do Debian Jessie com um ambiente chamado Blackbox, que é muito leve. Você pode fazer o download no link: https://www.bunsenlabs.org/. Você pode aprender mais sobre o Bunsenlabs, assistindo a meu vídeo de análise sobre o sistema: https://www.youtube.com/watch?v=M8G8UNdPDgY&t=208s

- **Microsoft Windows** (recomendo o Windows 7, mais leve em máquina virtual) - Para estudarmos compartilhamento de arquivos entre Windows e Linux usando o servidor SAMBA.

1.2 INTRODUÇÃO TEÓRICA

O Linux é um sistema operacional de código aberto, disponível para alterações, melhorias e novas implementações. Dizer "Linux" envolve um contexto que deve ser observado. É necessário compreender a diferença de "Distribuição GNU/Linux" e "kernel linux", ou em alguns casos, apenas: "Distro Linux".

Linux Distribuição

Linux Distribuição, Distribuição de Linux ou simplesmente Distro Linux é o empacotamento de diversos softwares, abertos (*open source*) ou não (proprietários) sobre o núcleo Linux. Podemos chamar de Distribuição Linux o Ubuntu, o Debian, o Centos, o Android e outras dezenas de sistemas operacionais existentes.

Você pode obter informação e realizar download de diversas distribuições de Linux, através do site: https://distrowatch.com/

Neste livro, como já mencionamos, usamos como base a distribuição Bunsenlabs que é uma customização do Debian Jessie.

GNU/Linux

Em 1984, antes mesmo de existir o núcleo Linux (kernel), Richard Stallman idealizou o desenvolvimento de um sistema operacional desprovido de amarras e travas de uso. Este sistema recebeu o nome de GNU, um acrônimo recursivo de GNU'Not Unix. Os desenvolvedores GNU criaram uma série de programas básicos para um sistema operacional funcional, como editores de texto e compiladores. Entretanto, havia um pedaço de código essencial, que ainda não tinha sido criado: o kernel.

No ano de 1991, Linus Torvalds publicou o Linux sob a mesma licença dos softwares GNU. Agrupando os softwares GNU com o kernel Linux, tínhamos o GNU/Linux. Até os dias de hoje, muitas pessoas discutem a forma com que se deve chamar uma Distro Linux. Pessoas que apoiam o movimento GNU, não abrem mão de dizer GNU/Linux, outros, porém, acreditam que as distribuições atuais não devem ser chamadas assim, porque possuem muitas ferramentas proprietárias; existem também aqueles que não se

preocupam com as filosofias e apenas usam o sistema.

Para quem deseja conhecer as distribuições que mais se adequam à filosofia GNU, recomendo que acessem: https://www.gnu.org/distros/free-distros.pt-br.html

Linux kernel

O núcleo (kernel) é o componente central de um sistema operacional, ele serve de ponte entre aplicativos e o processamento real de dados feito no nível de hardware. Tratando-se de kernel, temos dois modelos de grande utilização: microkernel e kernel monolítico.

Microkernel é o sistema operacional que possui apenas um núcleo que provê recursos mínimos necessários ao ambiente. Outras funcionalidades são oferecidas através de programas chamados servidores, que se localizam na *user space* - espaço do usuário.

O kernel monolítico é justamente o oposto do microkernel. Sua principal característica é permitir que funções como rede, vídeo e acesso a outros periféricos sejam possíveis através do *kernel space*. O Linux utiliza o modelo de kernel monolítico.

1.3 CONCLUSÃO

Agora que preparamos todo o nosso ambiente, vamos iniciar nossa jornada prática. Prepare um bom café, relaxe à frente de seu computador e vamos que vamos!

CAPÍTULO 2

PRIMEIROS PASSOS

O sistema operacional Bunsenlabs (Debian Jessie) já vem acompanhado da interface gráfica OpenBox após sua instalação. No entanto, este livro tem como objetivo, ensiná-lo a usar a poderosa linha de comando do Linux.

A linha de comando do GNU/Linux, também conhecida como *shell*, é muito poderosa. Podemos tratar tarefas repetitivas como uma "execução semanal de backup" através de scripts em shell, podemos estruturar programas complexos, integrados com banco de dados e muito mais.

Inicie o sistema Bunsenlabs (vou chamá-lo apenas de BL) através do Virtualbox. Caso ainda tenha dúvidas sobre como obter o sistema, recomendo minha videoaula: https://www.youtube.com/watch?v=M8G8UNdPDgY&t=208s

2.1 TERMINAL VIRTUAL

Terminal ou **console** são o teclado e o monitor conectados ao computador. O BL/Debian é um sistema operacional multiusuário, ou seja, suporta vários usuários conectados ao mesmo tempo, usando **terminais virtuais**. Um terminal virtual é uma segunda seção de trabalho completamente independente de outras e que

você poderá acessar através de um conjunto de teclas padrão.

Acessamos estes terminais virtuais segurando a tecla `< ALT >` e pressionando uma das teclas de `< F1 >` até `< F6 >`. O BL/Debian atribui `< F7 >` para a interface gráfica. Cada tecla tem função correspondente a um terminal do 1 ao 6.

O Linux possui mais de 63 terminais virtuais, mas inicialmente, estão disponíveis apenas 6. Quando você está no modo gráfico e deseja acessar algum dos terminais (modo texto), deverá apertar as teclas `< CTRL > + < ALT >` e o número correspondente do terminal virtual (1 a 6).

O modo gráfico do Linux é uma camada do sistema operacional que permite a você utilizar um gerenciador de janelas. No caso do BL/Debian, esta camada é possível graças a um servidor de X (gráfico) chamado Xorg, que está sendo substituído por outro de nome Wayland.

O Linux independe do servidor gráfico para funcionar. Aliás, em servidores, muitas vezes nem mesmo ele vai estar disponível. Por este motivo, vamos concentrar nosso estudo no modo texto do Linux, ou seja, sua linha de comando.

2.2 LOGON

Logon é a entrada do usuário, você deverá informar o nome de usuário e a senha, seja para o acesso como `root` ou com um usuário comum. Por padrão, quando você digita a senha não aparece nenhum retorno (os famosos asteriscos). O objetivo é evitar que um observador mais curioso seja capaz de contar a quantidade de caracteres.

Atuaremos sempre como usuário Administrador, ou seja, `root`. Por este motivo, é importante muita atenção na aplicação dos comandos, já que o `root` tem privilégios para fazer qualquer alteração no sistema, inclusive apagar arquivos importantes ao funcionamento do Linux.

2.3 INICIANDO PELO SHELL

O shell é o interpretador de comandos do Linux. Ele é a interface entre o usuário e o kernel do sistema. O shell padrão do Debian é o `bash`. Entretanto, existem outras interfaces, como `Csh`, `ksh`, `Zsh` e `tcsh`. A diferença entre eles é basicamente os atalhos e suas variáveis que definem funções próprias em cada um deles. Neste livro, nosso foco será o **bash**.

O shell bash interpreta toda a linha que você digita e imprime algum resultado. É comum, usuários fazerem referência ao shell, chamando-o de modo texto ou linha de comando.

O usuário no bash

Vale relembrar que, para diferenciar os usuários, o shell bash identifica o `root` com `#` (um sustenido) e o usuário comum com `$` (um cifrão). O bash, possui algumas funcionalidades, como: retornar comandos digitados anteriormente. Para navegar pelo histórico de comandos você pode usar as teclas: "Seta para cima" e/ou "Seta para baixo". O programa responsável por manter esta lista de comandos é o **history**. Você pode usar este comando como `root` ou usuário comum. Para isto, basta digitar:

```
# history
```

Outra funcionalidade importante é rolar a tela para baixo e para cima. Você poderá realizar esta ação usando as teclas `< Shift > + < PageUP >` e `< Shift > + < PageDown >`. Isto é útil para visualizarmos textos que rolam rapidamente para cima e saem do nosso campo de visão.

Para mudarmos do usuário comum para o usuário `root`, usamos:

```
$ su
```

Para mudarmos para um usuário específico, usamos:

```
$ su nome_de_usuario
```

Para mudarmos de usuário, carregando as suas variáveis locais, usamos:

```
$ su - nome_de_usuario
```

Existem dois comandos que lhe permitem saber quem é o usuário logado atualmente:

```
$ whoami
$ who am i
```

Caso você esteja logado com um usuário (exemplo: `aluno`) e faça a troca para o usuário `root` usando o comando `su`, a saída do comando `whoami` será `root`, já que ele mostra o usuário atual (que está logado neste momento).

No entanto, a saída do comando `who am i` será `aluno`, que é o usuário com que você se logou na máquina (primeiro a acessar o shell) - sendo que foi através dele que você acessou o usuário `root`.

2.4 CONFIGURANDO O TECLADO NO DEBIAN

O arquivo que possui informações sobre o teclado do BL/Debian encontra-se em:

```
/etc/default/keyboard
```

O arquivo a seguir tem uma configuração básica de um teclado português Brasil, ABNT-2:

```
# Check /usr/share/doc/keyboard-configuration/README.Debian for
# documentation on what to do after having modified this file.

# The following variables describe your keyboard and can have the
 same
# values as the XkbModel, XkbLayout, XkbVariant and XkbOptions op
tions
# in /etc/X11/xorg.conf.

XKBMODEL="pc105"
XKBLAYOUT="br"
XKBVARIANT="abnt2"
XKBOPTIONS=""

# If you don't want to use the XKB layout on the console, you can
# specify an alternative keymap.  Make sure it will be accessible
# before /usr is mounted.
# KMAP=/etc/console-setup/defkeymap.kmap.gz
```

Além de editar o arquivo, no BL/Debian você poderá usar o comando:

```
# dpkg-reconfigure keyboard-configuration
```

Após escolher o seu modelo de teclado, reinicie a configuração usando o comando:

```
# /etc/init.d/keyboard-setup restart
```

Para alterar de modo temporário a configuração do teclado (somente em uma sessão) você poderá usar o comando `loadkeys` , exemplo:

```
# lodkeys -d br-abnt2
```

Para teclado padrão americano:

```
# loadkeys -d us-acentos
```

Mouse no modo texto

Para instalar e utilizar o mouse no modo texto, utilize o comando:

```
# apt-get install gpm
```

2.5 HISTÓRICO DE COMANDOS

O shell bash armazena uma lista de todos os comandos que você já digitou. Isso facilita e muito a vida do administrador de sistemas Linux. Digite o comando:

```
$ history
```

Observe, na saída do comando, que as linhas são enumeradas. Algo como:

```
  1  xrandr
    2  clear
    3  df -h
    4  sudo apt-cache search pandoc
    5  sudo apt-get install pandoc
    6  su
    7  cat /etc/default/keyboard
    8  cat /etc/default/keyboard clear
    9  clear
   10  nano /etc/default/keyboard
```

```
11  xkill
12  history
```

Note que a linha dois tem o comando `clear`, que serve para limpar a tela. Caso eu queira executá-lo novamente, posso realizar através do comando:

```
$ !2
```

Aqui, o `!` (ponto de exclamação) é utilizado para chamar o comando que está na linha 2. Poderíamos usar qualquer linha, por exemplo:

```
$ !12
```

Caso não tenha a linha que escolheu, nenhum comando será mostrado.

2.6 O COMANDO FC

O comando `fc`, de *find command*, serve para você encontrar comandos dentro de um *range*, isto é, um intervalo. Exemplo:

```
$ fc -l 5 10
```

Nesse exemplo, estou exibindo apenas os comandos do histórico da linha 5 até a 10. Para visualizar os últimos 20 comandos:

```
$ find -l -20
```

Para visualizar todos os comandos desde o último que iniciem com `x`:

```
$ find -l x
```

2.7 LOGOUT

Logout é a saída do sistema. Realizamos através dos comandos:

```
$ logout
$ exit
$ <CTRL> + <D>
```

2.8 DESLIGANDO O COMPUTADOR

Somente usuários administradores como o `root` têm permissão para desligar o sistema. Podemos realizar esta tarefa usando alguns destes comandos:

```
# shutdown -h now
# halt
# poweroff
# init 0
```

Comando shutdown

O comando `shutdown` tem a seguinte sintaxe:

```
# shutdown <ação> <tempo>
```

A sintaxe `-h` do comando `shutdown` representa *halt*, que vem do comando em Assembly chamado `HTL` , que quer dizer "parada de processamento". (Assembly ou linguagem de montagem é uma notação legível para humanos do código de máquina.)

Podemos definir o tempo de desligamento em minutos, por exemplo:

```
# shutdown -h 12
```

Nesse caso, desligaremos o sistema em 12 minutos. Podemos

também, atribuir uma mensagem ao desligamento do sistema:

```
# shutdown -h 12 Vamos desligar em 12 minutos
```

Para cancelar o `shutdown` usamos o comando:

```
# shutdown -c
```

Para reiniciar o computador, trocamos `-h` por `-r` (*reboot*):

```
# shutdown -r 10 reiniciar em 10 minutos
```

2.9 REINICIANDO O COMPUTADOR

Reiniciar quer dizer "Iniciar novamente o sistema". Sempre evite usar os botões `Reset` ou `ON/OFF` de seu computador. Isso realiza uma abrupta suspensão de energia, o que pode causar problemas em seus discos.

No Linux, você pode usar os comandos: `reboot` ou `shutdown -r now` para reiniciar e desligar instantaneamente. Para reiniciar o computador daqui a 5 minutos (ou qualquer outro tempo), use o comando:

```
# shutdown -r 5
```

Para reiniciar o computador em 10 minutos oferecendo uma mensagem:

```
# shutdown -r 10 vamos reinicializar em breve
```

Para cancelar a reinicialização:

```
# shutdown -c
```

Para reinicializar imediatamente:

```
# shutdown -r now
```

2.10 ACESSANDO DIRETÓRIOS

Uma das tarefas mais cotidianas de quem usa Linux pela linha de comando é navegar pelos diretórios do sistema. Vamos iniciar, conhecendo o comando `pwd`, que informa o diretório atual:

```
# pwd
```

Outro comando muito comum é o `cd`, que é utilizado para que você mude de diretório:

```
# cd /home/juliano
```

Nesse exemplo, estamos acessando o diretório `/home/juliano`.

Para facilitar, podemos usar algumas facilidades ao comando `cd`, exemplo:

```
# cd ~
```

Usando o `~` (til), você terá acesso ao diretório `home` do seu usuário atual. Diretório `home` é o diretório de uso do usuário Linux. Neste local, o usuário tem todas as permissões. Caso você esteja logado como `root`, o diretório será `/root`, no entanto, caso esteja logado com um usuário comum, por padrão o diretório será `/home/nome-do-usuario`.

Estou logado com meu usuário `juliano`, vamos observar o retorno do comando:

```
$ cd ~
$ pwd
/home/juliano
```

Agora o retorno para o usuário `root`:

```
# cd ~
# pwd
/root
```

2.11 ACESSANDO A RAIZ

Para acessar o diretório raiz, que é identificado com a barra (`/`), usamos:

```
$ cd /
```

Para subir um nível na árvore de diretórios, podemos usar:

```
$ cd ..
```

Para subir dois níveis, usamos:

```
$ cd ../../
```

Para três níveis: `../../../` , e assim sucessivamente. Para retornar ao diretório anterior, podemos usar:

```
$ cd -
```

Para listar arquivos e diretórios, usamos o comando `ls` :

```
$ ls
```

Aqui, listamos os arquivos e diretórios do diretório atual. Para listar arquivos e diretórios em um *path* específico, use:

```
$ ls /etc/X11/
```

2.12 TECLAS DE ATALHOS

Conjunto de teclas	Definição
<	Apaga os caracteres à esquerda do cursor. < delete > - Apaga os

`backspace >`	caracteres à direita do cursor.
`< delete >`	Apaga os caracteres à direita do cursor.
`< end >`	Vai para o final da linha do comando.
`< ctrl > + < a >`	Move o cursor para o início da linha de comando.
`< ctrl > + < e >`	Move o cursor para o final da linha de comando.
`< ctrl > + < u >`	Recorta o que estiver à esquerda do cursor.
`< ctrl > + < y >`	Cola o conteúdo que foi recortado.
`< ctrl > + < l >`	Atalho para o comando de limpar a tela (*clear*).
`< ctrl > + < c >`	Abre uma nova linha de comando. Útil, quando temos uma linha "travada".
`< ctrl > + < d >`	Finaliza o shell.

2.13 CONCLUSÃO

Conhecer o básico sobre navegar pelos arquivos e diretórios do Linux é essencial para iniciarmos nossos estudos. Dedique-se um pouco a este capítulo, realizando testes em seu sistema. No começo, usar um ambiente de linha de comando pode parecer intimidador e até mesmo um modo antiquado de se fazer as coisas. Com o passar do tempo, você perceberá que o Linux é poderoso, justamente por ter um shell poderoso, que lhe permite uma administração rápida, através de arquivos de texto simples, o que é fácil de ser replicado a outros computadores. Coloque seu cinto de segurança, porque nossa nave já está partindo!

Capítulo 3

OBTENDO AJUDA

O Linux é um sistema operacional repleto de boa documentação. Mesmo sem acesso à internet, você terá muita informação local, sobre comandos e servidores através das páginas de manual, que estão disponíveis para você consultar.

3.1 A BUSCA PELO CONHECIMENTO

Qualquer administrador de sistemas Linux deve saber onde buscar informações para manter-se sempre atualizado. Neste capítulo, vamos aprender a consultar as documentações existentes e como buscar informações sobre o que precisamos.

A biblioteca de documentação do GNU/Linux é muito vasta. Antes de recorrermos à ajuda de outras pessoas, devemos lembrar que podemos ter a resposta dentro do próprio sistema, como veremos a seguir.

Mas é claro que o Linux cresce porque possui uma comunidade ativa. Esta comunidade não tem medo ou receio de compartilhar informação. Grande parte das pessoas que formam esta comunidade nunca ocultam seu *know-how*, ou seja, você pode perguntar à vontade, desde que saiba o que e onde perguntar.

3.2 AS MAN PAGES

Dentro do sistema, temos as famosas páginas de manual, denominadas de *Man Pages*. Esta documentação que possuímos no sistema também está disponível no site: http://www.tldp.org (*The Linux Documentation Project*).

Um diferencial deste site é que ele tem documentação em vários idiomas e formatos: PDF, HTML, TXT e outros.

3.3 FORMAS DE DOCUMENTAÇÃO

Existem diversas formas de se documentar um projeto, dentre elas temos os *How-to's*, os manuais e as documentações.

How-to's

Os How-to's são documentos que focam em uma necessidade específica. Exemplo: *Como montar um firewall*, *instalar uma impressora* etc. Normalmente esses documentos estão acompanhados de suas aplicações e/ou hardware. Os *How-to's* também são conhecidos como *cook-books* ou livros de receitas.

O diretório de How-to's do Linux é o `/usr/share/doc` .

Manuais

Diferente dos *How-to's*, os manuais não vão mostrar um passo a passo ou mesmo dar uma lista de afazeres. O principal objetivo do manual é nos mostrar como as funcionalidades daquele software podem ser usadas. Podemos executar o manual da maioria dos comandos GNU/Linux através do comando `man` .

Exemplo:

```
# man ls
```

3.4 COMANDOS DE AJUDA

O comando man

Sem dúvida, o comando `man` é um dos mais utilizados pelos administradores de sistema Linux. As páginas de manuais são divididas em níveis:

Nível	Descrição
man 1	Programas executáveis e comandos do shell
man 2	Chamadas de sistemas (funções providas pelo kernel)
man 3	Chamadas de bibliotecas (funções como bibliotecas do sistema)
man 4	Arquivos de dispositivos (Localizados normalmente no diretório `/dev`)
man 5	Arquivos de configurações e convenções
man 6	Jogos
man 7	Variados (incluindo pacotes de macros e convenções)
man 8	Comandos para a administração do sistema (normalmente usados pelo `root`)
man 9	Rotinas do kernel

Basicamente utilizamos o comando `man` da seguinte maneira:

```
$ man [ comando ]
```

Exemplo:

```
$ man passwd
```

Quando executamos esse comando, o comando `man` abre um arquivo que está compactado no diretório `/usr/share/man/man1`, que é o `passwd`. Outros níveis de manuais dependem do comando ou arquivo.

Você pode consultar quais manuais estão disponíveis dentro do próprio diretório do `man`:

```
$ ls /usr/share/man/
```

Dentro deste diretório, é possível ver todas as divisões dos manuais: os níveis, os idiomas e mais. Todos os níveis de manuais possuem sua determinada introdução, que pode ser vista com o comando:

```
$ man <nivel> intro
```

Exemplo:

```
$ man 5 intro
```

Por padrão, o Debian possui as páginas de manual em inglês. Podemos abrir em outro idioma com o comando:

```
$ man -L pt_BR [ comando ]
```

Recomendo que você instale o pacote de idioma:

```
# apt-get install manpages-pt
```

O comando help

O comando `help` provê uma ajuda de forma rápida. Ele é muito útil para saber quais opções podemos usar em um comando (shell). Exemplo:

```
# help
```

Caso desejamos visualizar a ajuda rápida para um comando, usamos:

```
# help cd
```

Para comandos externos. O `help` é um parâmetro, exemplo:

```
$ ls --help
```

O parâmetro `--help` pode ser utilizado em qualquer comando para ter uma consulta rápida dos parâmetros que ele pode nos oferecer.

O comando apropos

O comando `apropos` é utilizado quando não se sabe qual documentação acessar para um determinado assunto, mostrando as **man pages** que contêm a palavra-chave especificada.

A sintaxe básica deste comando é:

```
$ apropos [palavra-chave]
```

Exemplo:

```
$ apropos editor
```

O comando whatis

O comando `whatis` tem basicamente a mesma função do comando `apropos`, porém suas buscas são mais específicas. Ele

retorna apenas o manual de acordo com a palavra que você digitou, de forma exata. Exemplo:

```
$ whatis vim
```

O comando info

As *info pages* são como páginas de manuais, porém são utilizadas com navegação entre as páginas. Elas são acessadas com o comando `info`. Este é útil quando já sabemos o nome do comando e só queremos saber sua respectiva função.

Podemos navegar pelas páginas com as teclas `< n >` (next/próximo); `< p >` (previous/anterior); `< u >` (up/sobe um nível). Para sair do comando `info`, basta pressionar a tecla `< q >`.

```
$ info
```

Para exibir as informações somente de um determinado comando, usaremos a sintaxe:

```
$ info [comando]
```

Exemplo:

```
$ info vim
```

O comando whereis

O comando `whereis` é utilizado para mostrar a localização do binário do comando, do arquivo de configuração (caso exista), e a localização das páginas de manual. Exemplo:

```
$ whereis vim
vim: /usr/bin/vim.tiny /etc/vim /usr/share/vim /usr/share/man/man
1/vim.1.gz
```

Caso não exista o comando, nada será mostrado.

O comando which

O comando `which` é bem semelhante ao comando `whereis` , entretanto este só mostra a localização do binário do comando.

Para visualizar a localização do binário, usamos:

```
$ which comando
```

Exemplo:

```
$ which ls
/bin/ls
```

3.5 CONCLUSÃO

Usar as páginas de manual no terminal do Linux é mais comum do que você imagina, já que os comandos são recheados de opções. Um mesmo comando com uma opção diferente pode ter uma função diferente, por isto, na dúvida sempre utilize o comando **man**.

Outro recurso importante é a internet. Praticamente toda aplicação do Linux tem em sua página Web uma FAQ - Perguntas frequentes. As comunidades de usuários no mundo Linux são repletas e você encontrará boas pessoas, dispostas a estender as mãos para usuários iniciantes. Uma maneira rápida de começar sua rede de relacionamentos com a comunidade é se cadastrando em nossa rede: http://www.true.certificacoes.net.br

CAPÍTULO 4

COMANDOS GNU/LINUX

Comandos são introduções passadas ao sistema para executar uma determinada tarefa. No mundo GNU/Linux, o conceito de comandos é diferente do padrão MS-DOS. Um comando é qualquer arquivo executável, que pode ser ou não criado pelo usuário.

Entre as vantagens da linha de comando do GNU/Linux (shell) está a variedade de comandos que ele oferece, o que torna a administração do sistema rápida e eficaz.

O shell é o responsável pela interação entre o usuário e o sistema operacional, interpretando os comandos. É nele que os comandos são executados.

4.1 O COMANDO LS

O comando `ls` possui muitos parâmetros, veremos aqui os mais utilizados. A opção mais comum para este comando é o `-l` , que lista os arquivos ou diretórios de uma forma bem detalhada. Exemplo:

```
$ ls -l /
drwxr-xr-x   2 root root  4096 abr 25 06:15 bin
```

No exemplo anterior, pedimos para listar os arquivos e diretórios que se encontram na raiz do sistema (`/`). No meu caso, a primeira linha de saída do comando foi:

```
drwxr-xr-x   2 root root  4096 abr 25 06:15 bin
```

Vamos agora entender os campos que compõem esta linha, iniciando pelo primeiro caractere:

- **d** - Indica que se trata de um diretório

Poderia ser também:

- **l**: Indica que se trata de um link.
- **-**: Hífen, indica que se trata de um arquivo regular.
- **c**: Indica que o arquivo é um dispositivo de caractere (sem *buffer*).
- **b**: Indica que o arquivo é um dispositivo de bloco (com *buffer*).
- **u**: Sinônimo para o tipo `c` , indica que é um dispositivo de caractere (sem *buffer*).
- **s**: Indica que o arquivo é um *socket*.
- **p**: Indica que o arquivo é um FIFO, named pipe (pipe nomeado).

FIFO - SIGLA PARA *FIRST IN, FIRTS OUT* , que em inglês significa "primeiro a entrar, primeiro a sair". São amplamente utilizados para implementar filas de espera. Os elementos vão sendo colocados no final da fila e retirados por ordem de chegada. Pipes (`|`) são um exemplo de implementação de FIFO. **BUFFER** é uma região de memória temporária, usada para escrita e leitura de dados. Normalmente, os buffers são utilizados quando existe uma diferença entre a taxa em que os dados são recebidos e a taxa em que eles podem ser processados. **Socket** é um meio de comunicação por software entre um computador e outro. É uma combinação de um endereço IP, um protocolo e um número de porta de protocolo.

O campo `rwxr-xr-x` representa as permissões. Vamos estudá-las mais à frente neste livro.

O campo `root root` informa o dono do diretório ou arquivo, que neste caso é o `root` , e depois seu grupo primário, que também é `root` .

O campo `4096` indica o tamanho do arquivo ou diretório.

O campo `abr 25 06:15` informa a data e hora de criação do arquivo ou diretório.

O campo `bin` informa o nome do arquivo ou diretório, neste caso é o diretório `/bin` .

4.2 LS -A

O comando `ls -a` lista todos os arquivos, inclusive os arquivos ocultos. Exemplo:

```
# ls -a /root
.config      .gvfs     Público      xorg.conf.new
```

Na saída da minha primeira linha, observe que temos arquivos que se iniciam com ponto (`.`). Esses arquivos são ocultos. No Linux, arquivos e diretórios ocultos são iniciados por um ponto.

4.3 LS -R

O comando `ls -R` é utilizado para listar arquivos de forma recursiva, ou seja, listar também os subdiretórios que estão dentro de um determinado diretório. Exemplo:

```
$ ls -R /
/proc/5470/task/5508
```

Observe que ele listou o caminho completo de um determinado diretório (`/proc/5470/task/5508`).

4.4 CRIAR ARQUIVO

Para criar um arquivo, podemos simplesmente abrir um editor de texto e salvá-lo. Mas existem outras formas. A mais simples é usando o comando `touch` :

```
$ touch arquivo1
```

4.5 CURINGAS

No GNU/Linux, usamos *curingas* para especificar um ou mais arquivos e diretórios. Com este recurso, podemos: listar, copiar, remover e mover diversos arquivos e diretórios. São cinco tipos de curingas no Linux:

- `*` : Utilizado para um nome completo ou restante de um arquivo/diretório.
- `?` : Este curinga pode substituir uma ou mais letras em determinada posição.
- `!` : Exclui a operação.
- `[padrao]` : É utilizado para referência a uma faixa de caracteres de um arquivo/diretório.
- `{padrao}` : Expande e gera strings para pesquisa de padrões de um arquivo/diretório.

Vamos a um exemplo. Precisamos criar 3 arquivos chamados: `arq1`, `arq2` e `arq3` dentro do diretório `/home/juliano/`, usando o recurso *curinga*. O comando seria:

```
$ touch arq{1,2,3}
```

Segundo exemplo:

No diretório atual, temos diversos arquivos de imagens, com extensões do tipo: `.png`, `.jpeg`, `.gif`, `.pdf` entre outros formatos. Eu preciso listar somente as imagens que tenham a extensão `.png`, neste caso, uso o recurso *curinga*, realizando o comando:

```
$ ls *.png
```

Terceiro exemplo:

Podemos realizar exceções. No meu diretório atual, tenho os

arquivos: `arq1.txt` , `arq2.txt` e `arq3.txt` criados anteriormente. Quero listar todos eles, exceto o `arq2.txt` :

```
$ ls arq[!2].txt
```

4.6 CRIANDO DIRETÓRIOS

O comando `mkdir` é utilizado para criar um diretório no sistema. Um diretório é uma pasta onde você guarda seus arquivos. Exemplo:

```
$ mkdir videos
```

Podemos criar diretórios de forma recursiva, isto é, diretórios com subdiretórios, executando a sintaxe `-p` . Exemplo:

```
$ mkdir -p videos/terror
```

Com a opção `-p` , criamos toda uma estrutura de diretórios.

4.7 REMOVENDO ARQUIVOS/DIRETÓRIOS

O comando `rm` é utilizado para apagar arquivos e diretórios. Para apagar o arquivo `documento.txt` , por exemplo, usamos o comando:

```
$ rm documento.txt
```

Podemos usar a opção `-i` para forçar a confirmação de remoção do arquivo:

```
$ rm -i documento.txt
```

Para removermos diretórios, temos que usar a opção de recursividade `-r` ou `-R` . Exemplo:

```
$ rm -R /home/juliano/videos
```

Nesse caso, estamos removendo o diretório `videos`, que está localizado dentro dos diretórios `/home/juliano`.

Para removermos diretórios que estejam vazios, podemos usar o comando:

```
$ rmdir diretorio
```

O comando `rmdir` só permite a exclusão de diretórios vazios.

4.8 COPIAR ARQUIVOS/DIRETÓRIOS

O comando `cp` serve para fazer cópias de arquivos e diretórios. Para lidarmos com diretórios, temos que usar a opção de recursividade `-r` ou `-R`, assim como o comando `rm`.

Exemplos:

```
$ cp arquivo-origem arquivo-destino
$ cp arquivo-origem caminho/diretório-destino/
$ cp -R diretório-origem nome-do-destino
$ cp -R diretório-origem caminho/diretório-destino/
```

Uma opção interessante no comando `cp` é a `-p`, que permite que na cópia não se percam as informações do arquivo, tais como: data e hora de criação, donos e permissões. Exemplo:

```
$ cp -p arquivo-origem arquivo-destino
```

4.9 MOVENDO ARQUIVOS

O comando `mv` é utilizado tanto para mover arquivos e diretórios, quanto para renomeá-los. Exemplo:

```
$ mv arquivo1 arquivo2
```

Nesse caso, estamos renomeando `arquivo1` para `arquivo2`. Mais exemplos de uso do comando `mv`:

```
$ mv arquivo caminho/diretório-destino/
$ mv arquivo novo-nome
$ mv diretório novo-nome
$ mv diretório caminho/diretório-destino/
```

4.10 CONCLUSÃO

Deve-se ter muito cuidado ao executar tarefas de manipulação de arquivos com o usuário `root`. Como ele tem poder administrativo sobre todo o sistema, você poderia excluir arquivos importantes e corromper o seu sistema. Sempre observe os caminhos de destino no momento de mover ou copiar um arquivo e tenha atenção redobrada ao remover conteúdo.

Capítulo 5

FHS - HIERARQUIA DE ARQUIVOS

Desde que o GNU/Linux foi criado, muito se tem feito para seguir um padrão em relação à estrutura de diretórios. O primeiro esforço para a padronização de sistemas de arquivos foi o **FSSTND - Filesystem Standard**, lançado no ano de 1994.

Atualmente, é o **Filesystem Hierarchy Standard ou FHS** que determina o padrão da estrutura de diretórios do GNU/Linux. A versão atual deste padrão é a 2.3, anunciada em 29 de janeiro de 2004. O FHS é mantido pelo Free Standards Group, uma organização sem fins lucrativos formada por importantes empresas de hardware e software, como HP, Red Hat, IBM e Dell.

O padrão na estrutura de diretórios é muito importante. Ele ajuda a manter a compatibilidade entre as distribuições existentes no mercado, permitindo que qualquer software escrito para o GNU/Linux seja executado em qualquer distribuição desenvolvida.

5.1 ESTRUTURA DE DIRETÓRIOS

A estrutura de diretórios também é conhecida como "Árvore de diretórios" porque tem a forma de uma árvore. Um diretório é

como uma gaveta de armário. Cada gaveta guarda, normalmente, um tipo diferente de roupa, enquanto cada diretório guarda um tipo específico de arquivo.

O arquivo pode ser um texto, uma imagem, um vídeo, uma planilha entre outros. Os arquivos são identificados por nomes para que sejam localizados com facilidade.

Um detalhe importante a ser observado é que o GNU/Linux é *case sensitive*, isto é, ele diferencia letras maiúsculas nos arquivos e diretórios. Sendo assim, um arquivo chamado `Arquivo` é diferente de outro chamado `ARQUIVO` ou de outro chamado `arquivo`.

Para iniciarmos nosso estudo, executaremos o comando `ls`, que lista arquivos e diretórios. Partiremos do diretório raiz do sistema, de modo que através dele visualizaremos todos os diretórios do sistema, compreendendo assim a estrutura hierárquica.

```
$ ls /
```

Com isto, você verá a estrutura de diretórios do GNU/Linux. De acordo com o padrão FHS, esta estrutura deve conter obrigatoriamente 14 diretórios, que vamos especificar a seguir.

Diretório raiz

O diretório raiz do GNU/Linux é representado por uma `/` (barra). É no diretório raiz que ficam todos os demais diretórios do sistema. Estes diretórios, que vamos conhecer agora, são chamados de subdiretórios, pois estão dentro do diretório `/`.

Diretório `/bin`

O diretório `/bin` guarda os comandos essenciais para o funcionamento do sistema. Esse é um diretório público, sendo assim, os comandos que estão nele podem ser utilizados por qualquer usuário do sistema. Entre estes comandos, estão:

- `/bin/ls` : lista arquivos e diretórios;
- `/bin/cp` : copia arquivos e diretórios;
- `/bin/mkdir` : cria diretórios no sistema;
- `/bin/cat` : visualiza arquivos de texto.

Diretório `/boot`

No diretório `/boot` estão os arquivos necessários à inicialização do sistema, e o gerenciador de boot. O gerenciador de boot é um programa que lhe permite escolher os seus sistemas operacionais na inicialização do seu computador. Quando o computador é ligado, uma lista aparece para que você escolha entre diferentes versões do kernel Linux ou até mesmo outros sistemas operacionais que você possua, como o Microsoft Windows.

Diretório `/dev`

No diretório `/dev` ficam todos os **arquivos de dispositivos** (hardware do seu computador). O Linux faz a comunicação com estes periféricos por meio de links especiais que ficam armazenados neste diretório, facilitando o acesso a eles.

Diretório `/etc`

No diretório `/etc` estão os arquivos de configuração do

sistema, tais como: scripts de inicialização do sistema, tabela do sistema de arquivos, configuração padrão para logins dos usuários, entre outros.

Diretório `/lib`

No diretório `/lib` estão as bibliotecas compartilhadas e módulos do kernel. As bibliotecas são funções que podem ser utilizadas por vários programas.

Cada kernel tem seus próprios módulos, que ficam em: `/lib/modules/versao-do-seu-kernel/kernel` .

Diretório `/media`

O diretório `/media` é um ponto de montagem para dispositivos removíveis, tais como: HD, CD, DVD entre outros. Um ponto de montagem é um diretório que aponta para a partição física do disco, permitindo que você acesse seus arquivos e diretórios. Mais à frente no livro, abordaremos em detalhes a criação de partições e como acessá-las.

Diretório `/mnt`

O diretório `/mnt` é utilizado para montagem temporária dos sistemas, ou seja, para acessar um compartilhamento de arquivo em uma rede que depois não mais acessaremos, ou um HD externo que logo será desconectado do sistema.

Diretório `/opt`

O diretório `/opt` normalmente é utilizado por programas

proprietários ou que não fazem parte oficialmente da distribuição.

Diretório `/sbin`

O diretório `/sbin` guarda os comandos utilizados para inicializar, reparar, restaurar e/ou recuperar o sistema. Isso quer dizer que esse diretório contém comandos essenciais. Este diretório é de uso do administrador, `root` .

Diretório `/srv`

O diretório `/srv` guarda dados de serviços fornecidos pelo sistema, cuja aplicação é de alcance geral, ou seja, os dados não são específicos de um usuário. Por exemplo:

`/srv/ftp` (Servidor FTP) - Este diretório é usado para usuários anônimos que se conectam a um servidor FTP.

Diretório `/tmp`

O diretório `/tmp` serve para armazenamento de arquivos temporários.

Diretório `/usr`

O diretório `/usr` contém programas que não são essenciais ao sistema e que seguem o padrão GNU/Linux, como navegadores, gerenciadores de janelas entre outros.

Diretório `/var`

O diretório `/var` contém arquivos de dados de variáveis e arquivos de logs.

Diretório `/proc`

O `/proc` é um diretório virtual, ou seja, ele está armazenado na memória RAM e não no disco. Este diretório é mantido pelo kernel e, nele, encontramos configurações do sistema, dados estatísticos, dispositivos já montados, interrupções, endereços e estados das portas físicas, dados sobre as redes etc.

Diretório `/sys`

O diretório `/sys` é um primo do diretório `/proc` . Dentro dele, podemos encontrar quase o mesmo conteúdo que em `/proc` , mas de uma forma mais organizada para os administradores, já que ele estrutura melhor os subdiretórios. Para nós, administradores Linux, não existe diferença entre estes diretórios, é mais uma questão de gosto pessoal.

Diretório `/home` e `/root`

Os diretórios `/home` e `/root` podem estar disponíveis no sistema, mas não precisam obrigatoriamente possuir este nome. Por exemplo, o diretório `/home` poderia se chamar `/usuarios` , que não causaria nenhum impacto na estrutura do sistema.

No diretório `/home` temos os diretórios dos usuários, exemplo:

```
/home/juliano

/home/jaqueline
```

E dentro destes diretórios `juliano` ou `jaqueline` os arquivos pessoais deste usuário, tais como: músicas, filmes,

documentos de texto, planilhas, entre outros. No diretório `/root` temos os arquivos pessoais do usuário administrador do sistema.

5.2 CONCLUSÃO

A princípio, toda esta estrutura de diretórios do GNU/Linux parece ser bem confusa, principalmente para usuários que estão migrando do Microsoft Windows. No entanto, com o passar do tempo, você perceberá que o FHS é bem organizado e que facilita muito a vida dos desenvolvedores e dos administradores de sistema.

Capítulo 6

EDITOR DE TEXTO VIM

Praticamente toda a configuração de um sistema GNU/Linux é realizada através da edição direta de arquivos de configuração em modo texto. São diversas aplicações que podemos utilizar para editá-los, no entanto, como nosso foco é também se preparar para as certificações de Linux, utilizaremos o editor VIM.

Para definirmos no Debian qual será o nosso aplicativo de texto padrão, usamos o comando:

```
# update-alternatives --config editor
```

6.1 EDITOR DE TEXTO VIM

O **VI** é o editor básico do GNU/Linux, e está disponível em grande parte das distribuições de GNU/Linux. Hoje em dia, as distribuições usam uma versão mais completa e com mais recursos do que o VI, que é o **VIM = VI IMproved**.

Para instalar o VIM no Debian, utilize o comando:

```
# apt-get install vim
```

Para abrir o VIM digite:

```
# vim
```

Quando aberto, você poderá chamar a ajuda do VIM, digitando:

```
:help
```

Ou, simplesmente:

```
:h
```

6.2 COMO INTERPRETAR ATALHOS E COMANDOS

A tecla `< Control >` é representada na maioria dos manuais e no *help* do VIM pelo caractere `^` (circunflexo), ou seja, o atalho para `< Ctrl > + < L >` aparecerá assim:

```
^L
```

Vou criar como exemplo um arquivo vazio e abri-lo com o VIM:

```
# touch arquivo.txt
# vim arquivo.txt
```

Como estamos iniciando os nossos estudos com o editor VIM, é comum cometermos erros na edição de um texto. Para recarregar este arquivo que está sendo editado (`arquivo.txt`) sem as alterações, execute:

```
< esc >         Para sair do modo de edição
:e!            Recarrega o arquivo sem as alterações
```

Você também pode optar por fechar este arquivo, sem salvar suas modificações:

```
< esc >         Para sair do modo de edição
:q!            Sai do arquivo sem editá-lo
```

6.3 OS MODOS DO VIM

O VIM trabalha com três modos de operação. Vamos supor que temos o arquivo de nome `texto.txt` . Podemos abri-lo usando o comando:

```
$ vim texto.txt
```

Ele abre no modo de **visualização**. Quando pressionamos a tecla `< i >` ou `< insert >` ele abre o modo de **inserção**. Quando você pressiona a tecla `< esc >` ele volta para o modo de **comandos**. Neste modo, você digita um comando colocando dois pontos à frente. Exemplo:

```
:help
```

Para acessar o modo de visualização novamente, pressione a tecla `< v >` .

Resumindo:

- **O modo de inserção** é o modo de edição. Este modo é como o bloco de notas do Windows.
- **O modo de comandos** é onde você vai formatar o texto (deletar, copiar e colar, substituir etc.).
- **O modo visual** é onde você faz seleção de grandes blocos de texto.

6.4 ABRINDO CAMINHOS DENTRO DE UM ARQUIVO

Caso esteja visualizando um arquivo no VIM e ele tenha alguma linha de referência a outro arquivo dentro do sistema, você

pode abri-la, parando sobre a linha do arquivo, digitando `< esc >` e depois as teclas `< ctrl > + < w > + < f >`.

Por exemplo, considere que está lendo um arquivo e encontra no meio dele a linha:

```
O arquivo /etc/passwd contém dados do usuário.
```

Use a tecla `< esc >`, deixe o cursor sobre a referência `/etc/passwd`, pressione as teclas: `< ctrl > + < w > + < f >` e o VIM abrirá este arquivo na parte superior. Para fechar, utilize:

```
< esc >
:q
```

Abrindo arquivo em uma linha específica

Imagine a seguinte situação: você abre um arquivo de configuração como o do proxy `squid3` que tem mais de 7.000 mil linhas, e precisa encontrar um determinado comando. Para não ficar um bom tempo navegando sobre o arquivo, utilize:

```
$ grep -n "palavra" arquivo
```

Após a execução desse comando, será exibido no terminal todas as linhas com o termo pesquisado:

```
$ grep -n "juliano" /etc/passwd
27:juliano:x:1000:1000:Juliano Ramos,,,:/home/juliano:/bin/bash
```

Agora que temos o número da linha, no caso do comando executado, linha 27, podemos abrir este arquivo, com:

```
$ vim +27 /etc/passwd
```

Deste modo, já abrimos o arquivo `/etc/passwd` com o VIM direto na linha 27.

Alternando entre arquivos

Agora, se você está digitando um script de backup e alguns parâmetros de configuração estão em outro arquivo, para não precisar fechar o editor, execute o comando a seguir:

```
:e /caminho/arquivo
```

Este comando vai permitir que você abra novos arquivos no VIM sem fechar o editor. Quando estamos editando mais de um arquivo, usamos as teclas: `< ctrl > + < 6 >` para alternar entre eles.

Movimentando-se sobre o texto

Estando no modo de visualização do editor VIM, o melhor modo de se movimentar pelo texto é com as teclas: `< h >`, `< j >`, `< k >` e `< l >`. Pode-se pensar que pelas teclas direcionais é mais fácil, e elas funcionam, de fato, mas depois de um tempo usando as letras percebe-se que elas estão mais próximas de outras letras que são comandos. Vejamos a seguir suas funções:

- `< h >`: esquerda
- `< j >`: abaixo
- `< k >`: acima
- `< l >`: direita

Agora, vamos aprender algumas outras opções, notando que são *case sensitive* (há diferença entre maiúscula e minúscula).

Comando	Definição
`< G >`	Vai para o final do arquivo.
`< gg >`	Volta ao topo.

< v >	Para mover ao final de uma palavra.
< ge >	Para mover ao final de uma palavra voltando ao cursor.
< w >	Para mover até o começo da próxima palavra.
< b >	Para mover até o começo de uma palavra voltando ao cursor.
< f [caractere] >	Para mover ao próximo caractere específico.
< 0 >	Para mover ao começo da linha.
< $ >	Para mover ao final de uma linha.
< } >	Para pular ao final de um parágrafo.
< ctrl > + < G >	Para saber sua posição no arquivo.
[número da linha] + < G >	Para seguir para a linha especificada.

O VIM mantém um histórico de alterações. Para movimentar-se entre as alterações, use:

- < g > + < ; > : segue para a última alteração;
- < g > + < , > : segue para a próxima da lista.

Inserindo texto

Para sairmos do modo de visualização do VIM e começarmos a inserir nosso texto, usamos as seguintes teclas:

Comando	Definição
< i >	Entra no modo de inserção no local onde o cursor se encontra.
< I >	Entra no modo de inserção no começo da linha.
< a >	Entra no modo de inserção na frente de onde o cursor se encontra.
< A >	Entra no modo de inserção no final da linha.

< o >	Adiciona uma linha acima e entra em modo de inserção.
< ESC >	Volta ao modo normal/comando.
< v >	Entra no modo visualização.

Deletando texto

Estando no modo de visualização, podemos usar as teclas:

Comando	Definição
< x >	Deleta o caractere onde o cursor estiver.
< X >	Deleta o caractere atrás do cursor.
< d > + < w >	Deleta a partir do cursor até o começo da próxima palavra.
< d > + < $ >	Deleta o cursor até o final da linha.
< d > + < d >	Deleta a linha inteira.
< C >	Deleta de onde o cursor estiver até o final da linha.
< c > + < e >	Deleta do cursor até o final da palavra.

Copiar e colar

Comando	Definição
< y >	Copia a palavra.
< y > + < y >	Copia a linha inteira.
< p >	Cola.

O melhor modo de se copiar algo no VIM é navegando até a linha ou palavra que se deseja, entrando no modo visual (tecla < v >) e selecionando a linha com as teclas de movimentação. Após selecionar, use a tecla < y > para copiar, saia do modo de visualização com < esc > , movimente-se até o local de colar e

pressione `< p >`.

Recortar e colar

Para recortar e colar, use algum comando de exclusão (`x` , `dd` , `dw` ou `d$`), movimente-se até o local onde deseja colar e use a tecla `< p >`.

O último texto/palavra que você deletou será inserido após o cursor.

Substituição

Um recurso muito útil em qualquer editor de texto é a substituição de palavras. No VIM, isto é realizado no modo de comandos. Caso esteja no modo de visualização ou inserção, pressione a tecla `< esc >` e depois `< : >` , seguido de algum comando adiante:

Comando	Definição
`s/antiga/nova`	Substitui a ocorrência de antiga para nova na mesma linha.
`% s/antiga/nova`	Substitui em todo o arquivo.
`% s/antiga/nova/gc`	Substitui em todo o arquivo, mas solicita confirmação em cada ocorrência.

Buscando texto dentro do arquivo

Para buscar texto dentro do VIM, no modo de comandos, utilizamos sempre à frente do dois pontos (`< : >`) os comandos adiante:

Comando	Definição

`/termo-a-ser-pesquisado`	Pesquisa para baixo do arquivo.
`?/termo`	Pesquisa para cima do arquivo.
`//`	Busca o último termo pesquisado.

Ao entrar no modo de busca, o VIM deixa a palavra pesquisada na barra inferior, então, podemos usar:

Comando	Definição
`< n >`	Para a próxima ocorrência.
`< N >`	Para a ocorrência anterior.

6.5 CONCLUSÃO

Praticamente todos os servidores que rodam sobre o Linux utilizam arquivos de configuração em texto puro. Como não usamos interface gráfica em um servidor Linux, conhecer o VIM é fundamental. O VIM é cobrado nos exames de certificação LPIC-1 e LTC (*Linux True Certificate*).

Capítulo 7

PRIMEIROS PASSOS NO SHELL SCRIPT

O termo técnico "shell", em computação, é considerado genericamente a camada externa entre o usuário e o kernel (núcleo de um sistema operacional). É mais comumente utilizado para se referir aos programas de sistemas do tipo UNIX como o GNU/Linux, isto porque estes sistemas operacionais trabalham muito bem através da linha de comando, o que chamamos de CLI - Interface de linha de comando.

O Microsoft Windows atua com a GUI - Interface Gráfica do usuário, por este motivo, poucos usuários deste sistema operacional conhecem ou usam um shell. Em livre tradução para português, shell pode significar "concha" ou "casca".

O primeiro shell

O primeiro shell unix foi criado por Ken Thompson e chamado de `sh` na década de 70. Praticamente todas as distribuições de GNU/Linux possuem o `sh` no sistema, no entanto, o interpretador `bash` é que se tornou padrão.

O *shell unix* permite muito mais do que a execução dos

comandos que digitamos no prompt. É possível criar programas poderosos usando as sintaxes do shell assim como scripts que facilitam a vida do administrador de sistema, tornando mais simples as execuções de tarefas repetitivas no dia a dia.

Um script nada mais é do que um algoritmo projetado para realizar uma determinada tarefa. No mundo Unix e Linux, chamamos os scripts criados no shell de **shell script**. Mas o poder do shell vai muito além de scripts para a automatização do sistema. Você pode criar programas poderosos com ele, como softwares de gestão empresarial.

Criação de um shell script

Você pode criar seu script em shell usando qualquer editor de texto puro. Como no capítulo anterior abordamos o editor VIM, vamos usá-lo para criar nossos programas.

No Linux, extensões de arquivos não é algo essencial. Podemos criar um shell script com ou sem uma extensão, no entanto, quando colocamos a extensão (`.sh`) no nome do arquivo, o editor VIM habilitará as sintaxes coloridas, que vão facilitar na escrita do script.

Exemplo:

```
#$ vim olamundo.sh
```

As primeiras linhas de um shell script devem ser sua referência ao interpretador de comandos, ou seja, o caminho para sua execução. Como utilizaremos o `bash` , o caminho é representado como:

```
#!/bin/bash
```

Para tornar isso mais interessante e divertido, vamos fazer o nosso script retornar uma mensagem de "Olá mundo" no prompt:

```
#!/bin/bash
echo "Olá mundo"
```

Salve este arquivo e feche-o. Observe que só temos duas linhas neste script. Na primeira, fazemos a referência à localização do bash e na segunda usamos o comando `echo`, que imprime o conteúdo que está entre as aspas no prompt.

Concedendo permissões ao arquivo

Para executarmos um script em shell é necessário que ele tenha permissão de execução. Para isto, utilizamos o comando: `chmod`.

Atribuindo a permissão de execução ao arquivo `olamundo.sh`, temos:

```
#$ chmod +x olamundo.sh
```

Executando um script em shell

Agora que atribuímos a permissão de execução ao nosso script, podemos executá-lo de dois modos:

```
#$ bash olamundo.sh
#$ ./olamundo.sh
```

O resultado será:

```
Olá mundo
```

Que é a saída do comando `echo` presente no script.

Elaborando melhor nosso script

Vamos começar a nos divertir um pouco mais com o shell. Seu poder é ilimitado! Imagine que, de tempos em tempos, você precise visualizar algumas informações do sistema, como: nome de usuário logado, data e tempo em que o servidor está ligado entre outras coisas. Para fazer estas tarefas, no mínimo você terá que digitar três ou quatro comandos. Criando um script em shell, podemos facilitar este processo, executando apenas um comando e, ainda, tendo um retorno visual mais bonito.

No exemplo a seguir, vou criar um script bem simples que vai retornar ao administrador informações sobre o usuário logado, data e tempo em que o computador está ligado e o diretório atual em que o usuário está.

> É muito importante para o seu aprendizado, não somente em shell script mas em qualquer outra linguagem de programação, que você sempre reproduza os códigos. Pense nesta tarefa como um exercício de academia: no começo é bem difícil de se realizar, mas com o tempo e persistência, eles vão se tornando simples, ao ponto de você tentar coisas novas.

```
#!/bin/bash
echo "Seu nome de usuário é:"
whoami
echo "Informação de hora e tempo em que o computador está ligado:
"
uptime
echo "O script está no diretório:"
pwd
```

Salve este arquivo. Coloque a permissão de execução e execute-o.

Comentários no script

Os comentários dentro de um código-fonte são de grande importância, principalmente quando outros programadores vão analisá-lo. Funcionam como uma espécie de instrução para um determinado trecho de código. Exemplo de código com comentário:

```
#!/bin/bash
echo "Seu nome de usuário é:"
# O comando whoami informa o nome do usuário.
whoami

echo "Seu diretório atual é:"
# O comando pwd informa o diretório atual
pwd
```

Observe que os comentários são marcados com um # (sustenido). Estas linhas não são executadas no script e só podem ser vistas ao abrir o código-fonte, ou seja, o arquivo do shell.

7.1 DECLARANDO E USANDO VARIÁVEIS

Diferente de outras linguagens de programação, no shell script, não precisamos definir o tipo da variável. Para declará-las, basta seguir a sintaxe:

```
nome-da-variavel=valor
```

O nome da variável, pode começar com letra maiúscula ou minúscula além de underscore (_) e não pode conter espaço.

Exemplo de variáveis dentro de um script em shell:

```
#!/bin/bash
site=www.certificacoes.net.br
meu_numero_favorito=13
_cidade="São Paulo"

echo "Eu acesso todos os dias o site $site"
echo "Meu número favorito é $meu_numero"
echo "Minha cidade natal é $_cidade"
```

Observe que sempre usamos o símbolo de cifrão (`$`) para ler o conteúdo de uma variável. Para imprimir o nome de uma variável em vez do seu conteúdo, usamos à frente uma barra invertida, `\` :

```
#!/bin/bash
nome=fernanda
echo "O nome da variável é \$nome"
```

Armazenar saída de comando em variável

Podemos armazenar a saída de um comando em uma variável. Isto é muito útil quando usamos o mesmo comando diversas vezes dentro de um script, principalmente quando este comando é longo para ser digitado. Veja o exemplo:

```
#!/bin/bash
upgrade=`apt-cache update && apt-get upgrade`
echo $upgrade
```

Observe que usamos o acento de crase, isto diferencia o conteúdo com sendo um comando, em vez de um texto. A partir deste momento, dentro deste script, em vez de digitarmos o comando `apt-cache update && apt-get upgrade` vamos digitar apenas: `echo $upgrade` .

Capturando a entrada de dados do usuário

O shell script é tão poderoso que permite que você faça interação com o usuário do script. Podemos criar uma agenda, por exemplo, em que o usuário terá que fornecer alguns dados como: nome, telefone, endereço e e-mail. Para que isso seja possível, o shell script vai ler o que o usuário digitou e armazenar em uma variável por meio do comando `read` . Exemplo:

```
read nome-da-variavel
```

Vejamos um exemplo prático:

```
#!/bin/bash
Qual o nome de sua música favorita?
read nome-da-musica

echo "Você gosta de ouvir $nome-da-musica"
```

7.2 COMANDO DE SELEÇÃO OU TOMADA DE DECISÃO

Para deixar um script mais robusto, podemos usar comandos de tomada de decisão. Para exemplificar, vamos pensar em um jogo de videogame: o personagem precisa encontrar uma chave para abrir o baú. Se ele tem a chave, o baú abre, se não tem, o baú não abre. Observe a condição **se** que utilizei no exemplo. Esta condição é muito comum em praticamente todas as linguagens de programação, incluindo o shell script. Vejamos como ela funciona:

```
if [condição]
then
Ações
fi
```

Onde:

- **condição** - Teste que, se verdadeiro, passará o controle

para o bloco dentro do `then` ;

- **Ações** - Comandos a serem executados se o resultado de **condição** for verdadeiro.

É muito comum o desenvolvedor esquecer de fechar o `if` . Lembre-se sempre de que, para cada `if` aberto, você precisa fechá-lo com o comando `fi` .

Antes de continuarmos com um exemplo prático, vamos conhecer uma lista dos comandos mais utilizados para se testar uma condicional:

- string1 `=` string2: testa se string1 e string2 são idênticas;
- string1 `!=` string2: testa se string1 e string2 são diferentes;
- inteiro1 `-eq` inteiro2: testa se inteiro1 possui o mesmo valor que inteiro2;
- inteiro1 `-ne` inteiro2: testa se inteiro1 não possui o mesmo valor que inteiro2;
- inteiro1 `-gt` inteiro2: testa se inteiro1 é maior que inteiro2
- inteiro1 `-ge` inteiro2: testa se inteiro1 é maior ou igual a inteiro2
- inteiro1 `-lt` inteiro2: testa se inteiro1 é menor que inteiro2
- inteiro1 `-le` inteiro2: testa se inteiro1 é menor ou igual a inteiro2

Outras opções de uso comum:

- `e nome_do_arquivo` : verifica se um arquivo existe;
- `d nome_do_arquivo` : verifica se `nome_do_arquivo` é um diretório;

- `f nome_do_arquivo` : verifica se `nome_do_arquivo` é um arquivo comum;

Exemplo prático:

```
#!/bin/bash
echo "Digite um número qualquer:"
read numero;

if[ "$numero" -gt 20 ];
then
echo "Este número é maior que 20!"
fi
```

O comando `else`

Existe a possibilidade de tratarmos a tomada de decisão quando ocorre falha. Para isto, usamos o comando `else` , cuja sintaxe é:

```
if [condição]
then
Ações_1
else
Ações_2
fi
```

Onde:

- **condição:** teste que, se verdadeiro, passará o controle para o bloco dentro de `then` ;
- **Ações_1**: comandos a serem executados se o resultado de **condição** for verdadeiro;
- **Ações_2:** comandos a serem executados se o resultado de **condição** for falso.

No exemplo a seguir, verificaremos se um número digitado

pelo usuário é positivo ou negativo.

```
#!/bin/bash
echo "Digite um número:"
read numero;
if ["$numero" -ge 0]
then
    echo "O número $numero é positivo"

else
    echo "O número $numero é negativo"
fi
```

O comando `elif`

Há casos em que temos mais de uma condição a ser testada, e todas são correlacionadas. Para isto, usamos o comando `elif`, cuja sintaxe é:

```
if [ condição_1 ];
then
    Ações_1

elif [ condição_2 ];
then
    Ações_2

elif [ condição_3 ];
then
    Ações_3

    fi
```

Exemplo:

```
#!/bin/bash
echo "Selecione uma opção:"
echo "1 - Exibir a hora do sistema:"
echo "2 - Mostrar o diretório atual:"
echo "3 - Exibir uma mensagem qualquer:"
read opcao;
```

```
if [ $opcao == "1"];
then
data=`date`
echo "$data"

elif [ $opcao == "2"];
then
diretorio=`pwd`
echo "$diretorio"

elif [ $opcao == "3"];
then
echo "Digite seu nome:"
read nome;
echo "Bem-vindo $nome";
fi
```

O comando `case`

O comando `case` tem a mesma funcionalidade do `if...then..elif` , com a diferença de sua sintaxe ser mais compacta:

```
case VARIAVEL in
caso_1)
Acões_1
;;

caso_2)
Ações_2
;;

caso_3)
Ações_3
;;
esac
```

Vamos modificar o exemplo anterior, usando o `case` :

```
#!/bin/bash
echo "Selecione uma opção:"
echo "1 - Exibir a hora do sistema:"
```

```
echo "2 - Mostrar o diretório atual:"
echo "3 - Exibir uma mensagem qualquer:"
read opcao;

case $opcao in
"1")
data=`date`
echo $date
;;

"2")
diretorio=`pwd`
echo $diretorio
;;

"3")
echo "Informe o seu nome:"
read nome;
echo "Bem-vindo $nome"
;;
esac
```

7.3 LOOPS CONDICIONAIS

Loop `for`

Um loop é uma ação repetitiva que só é interrompida quando uma condição é satisfeita.

Vejamos agora um exemplo prático do comando de loop `for` :

```
#!/bin/bash
echo "Testando o comando seq"
for numero in $(seq 1 100);
do
echo "$numero"
done
```

Neste exemplo, faremos uma contagem de 1 a 100 através do comando `seq` .

Loop `while`

Enquanto o loop `for` é ideal para quando sabemos até quando contar, o loop `while` é bom para quando não temos esta noção, mas sabemos de uma condição que deverá ser atendida para que o laço termine. Exemplo:

```
#!/bin/bash
echo "Informe o que desejar, -1 para sair"
read dado;
while [ $dado != "-1"];
do
echo "Você digitou $dado"
read dado;
done
```

7.4 FUNÇÕES

O uso de funções é imprescindível para separar, organizar e estruturar a lógica de qualquer algoritmo. Sua sintaxe é bem simples:

```
nome_funcao{}
{
  Ações
}
```

Funções podem chamar outras funções existentes no script, simplesmente escrevendo-se o nome dela, como vemos no exemplo a seguir:

```
#!/bin/bash
main()
{
  echo "Escolha uma opção:"
  echo "Listar o diretório tmp."
  echo "Ver o diretório atual."
```

```
 read opcao;
 case $opcao in
 "1")
 listar_diretorio
 ;;
 "2")
ver_diretorio
 ;;
 esac
 }

 listar_diretorio()
 {
   echo "Listar diretório"
   listar=`ls /tmp`
   echo $listar
     }

 ver_diretorio()
 {
   echo "Ver diretório"
   ver=`pwd`
   echo $pwd


 }
 main
```

Nesse exemplo, a função principal chama-se `main` , declarada quando colocamos `main()` no começo do script. Dentro do comando `case` , há duas novas funções: `listar_diretório` e `ver_diretorio` . Para executar estas funções, mais abaixo, executamos os comandos `listar_diretorio()` e depois `ver_diretorio()` . Dentro de cada uma das duas funções, note que existe conteúdo.

Estes conteúdos formam blocos de código organizados, de forma que o programador consiga identificar erros dentro do seu script de forma rápida. Ainda seguindo o exemplo anterior, digamos que a opção de visualizar um diretório não esteja

funcionando. Em vez de olharmos todo o código-fonte do nosso script, seguimos para o bloco da função chamada `ver_diretorio` , desta forma, nosso script fica profissional.

Lembre-se sempre de chamar a função principal (neste caso, `main`) no final do seu script, do contrário, nada acontecerá quando você executar.

7.5 CONCLUSÃO

O shell script pode ser uma grande ferramenta para os administradores de sistema. Com esta pequena base, você já poderá criar scripts simples como a execução de um backup, configuração automática de uma interface de rede, entre outras tarefas cotidianas, que você aprenderá nos próximos capítulos deste livro.

CAPÍTULO 8

INTRODUÇÃO A REDES

Neste capítulo, vamos aprender conceitos básicos e fundamentais sobre redes de computadores. Um administrador de sistemas Linux deve compreender o funcionamento do IP, máscara de rede, gateway e DNS. Ao final do capítulo, você será capaz de configurar manualmente uma interface de rede. Vamos iniciar conhecendo os protocolos TCP/IP.

8.1 OS PROTOCOLOS TCP/IP

O protocolo TCP (*Transmission Control Protocol*) é orientado a conexões e transporta informações por meio de um processo chamado *Handshake*. O Handshake, ou aperto de mão, é o processo pelo qual duas máquinas afirmam uma a outra que a reconheceu e está pronta para iniciar a comunicação. É utilizado em diversos protocolos, como o FTP (*File Transfer Protocol*), que é uma forma de transferir arquivos pela rede, e o HTTP (*Hypertext Transfer Protocol*), que é um protocolo de comunicação para a troca de hipertexto sendo a base da World Wide Web.

O IP (*Internet Protocol*) é um protocolo de comunicação usado entre todas as máquinas em rede para o encaminhamento dos dados.

Entendendo o IPv4

O endereçamento IPv4 - versão 4 é composto por 4 octetos e uma máscara que determina quantos endereços são destinados a host, que são: computadores, impressoras, dispositivos móveis etc., e quantos endereços são destinados a uma rede local - também conhecida como LAN. O Linux não difere de outros sistemas operacionais, portanto, para termos acesso a uma rede LAN e/ou internet é necessário configurarmos um endereço IP.

O endereço IP está presente em todos os computadores, mesmo os que não possuem conexão com a internet, através de uma interface lógica, chamada de *loopback* ou "lo" cujo endereço IP é sempre 127.0.0.1 e que deve sempre estar configurado de maneira correta.

Mas por que precisamos ter este endereço?

Algumas aplicações utilizam este endereço de loopback (127.0.0.1) para comunicação com outras aplicações internas. Além disso, você pode desenvolver suas páginas para a Web e testá-las localmente, caso tenha um servidor Web instalado em seu sistema, apontando no browser para o endereço de loopback.

A internet é totalmente endereçada por números IPs. Quando você coloca um nome de um site no seu navegador Web, não está fazendo nada mais que apontando seu computador para uma máquina na grande rede.

Para obter o número IP do site que você está navegando, você pode usar o comando `ping` . Exemplo:

```
ping www.certificacoes.net.br
```

Observe a saída do comando:

```
PING certificacoes.net.br (169.57.129.229) 56(84) bytes of data.
64 bytes from nuvem28br.hoteldaweb.com.br (169.57.129.229): icmp_
seq=1 ttl=52 time=38.9 ms
```

Perceba que o IP do site `certificacoes.net.br` é `169.57.129.229` , que é o endereço de um servidor que hospeda o site visitado. O seu navegador consegue transformar o endereço digitado no número IP por causa de um servidor chamado DNS (*Domain Name Server*), ou Servidor de nomes.

Apesar de o DNS ter uma função muito útil, por facilitar o acesso a internet e permitir que a navegação seja feita através de nomes e não de números, ele não é essencial para o funcionamento da internet. Poderíamos acessar os endereços diretamente pelo número IP.

Se você está acessando sites pelo seu navegador, saiba que seu sistema operacional está com o DNS corretamente configurado. Um teste rápido quando um usuário alega que está sem conexão com a internet é pedir-lhe que realize o comando ping em um IP da internet. Exemplo:

```
#$ ping 8.8.8.8
```

Este endereço (8.8.8.8) é de um servidor DNS do Google. Caso o resultado do comando seja positivo, como:

```
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=48 time=75.5 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=48 time=75.5 ms
```

Já podemos concluir que a internet está funcionando, porém o DNS está configurado de modo errado. Normalmente utilizamos servidores DNS externos, que são oferecidos pelo nosso serviço de

internet, porém, é possível termos o nosso próprio servidor DNS.

O gateway da rede

No inglês, *gateway* pode ser traduzido como "porta de entrada"; tratando-se de rede de computadores, a definição é praticamente a mesma. O gateway é um host que tem a missão de encaminhar você para o mundo externo, além de sua LAN, que definimos como WAN (*Wide Area Network*), uma rede maior como um campus de uma universidade ou a própria internet.

Sendo assim, basicamente precisamos realizar três configurações em nosso sistema para acessar a internet:

1. Configurar um endereço IP - Compatível com nossa rede, a LAN.
2. Configurar um endereço DNS - Para resolver os nomes dos sites.
3. Configurar um gateway - Para termos acesso à rede externa, além da LAN.

Configurando o IP

O primeiro passo para configurarmos nossa interface de rede (placa de rede) é saber qual nome foi atribuído a ela pelo nosso sistema. Para isto, podemos usar o comando:

```
#$ ifconfig -a
```

Caso este comando não esteja disponível em sua distribuição de GNU/Linux, instale o pacote (software) **net-tools** com o comando:

```
$ apt-get install net-tools
```

Após instalar o pacote, execute novamente o comando `ifconfig -a` . O resultado será algo como:

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.75  netmask 255.255.255.0  broadcast 192.1
68.1.255
        inet6 fe80::d250:99ff:feae:e646  prefixlen 64  scopeid 0x
20<link>
        inet6 2804:431:9715:b324:d250:99ff:feae:e646  prefixlen 6
4  scopeid 0x0<global>
        inet6 fd37:267c:7d7a:1:d250:99ff:feae:e646  prefixlen 64
 scopeid 0x0<global>
        ether d0:50:99:ae:e6:46  txqueuelen 1000  (Ethernet)
        RX packets 667986  bytes 343008480 (327.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 831765  bytes 760666770 (725.4 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions
0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1  (Loopback Local)
        RX packets 86  bytes 5118 (4.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 86  bytes 5118 (4.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions
0

wlan0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 7a:f8:cf:d0:af:d0  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions
0
```

No meu caso, tenho três interfaces de rede:

- **eth0** - Minha placa de rede ethernet.
- **lo** - Endereço de loopback .
- **Wlan0** - Minha interface sem fio, wireless.

OBSERVAÇÃO:

O Debian 9 tem atribuído à primeira interface ethernet o nome `enp0s3` mas este nome pode mudar de distribuição para distribuição de Linux; no meu caso o nome é `eth0` .

Máscara de sub-rede

Além de configurarmos um endereço IP, é necessário configurarmos sua máscara de sub-rede, também conhecida como *netmask*. Diferente do endereço IP, que é formado por valores entre 0 e 255, a máscara de sub-rede é normalmente formada por apenas dois valores: 0 e 255, como em: 255.255.0.0 ou 255.0.0.0, onde o valor 255 indica a parte do endereço IP referente à rede e o valor 0 indica a parte do endereço IP referente ao host.

Os endereços IPv4 possuem 3 classes básicas que chamamos de: classe A, classe B e classe C. Elas definem a quantidade de hosts possíveis, ou seja, quantos números IPs poderemos usar. Para cada classe de IP temos uma máscara de rede padrão:

Classe	Início	Fim	Netmask	Notação
A	1.0.0.1	126.255.255.254	255.0.0.0	/8
B	128.0.0.1	191.255.255.254	255.255.0.0	/16
C	192.0.0.1	223.255.255.254	255.255.255.0	/24

No GNU/Linux temos dois tipos de configuração de rede: dinâmica e estática.

8.2 CONFIGURAÇÃO DINÂMICA

Chamamos de configuração dinâmica de rede uma configuração que vai se perder ao reiniciarmos o sistema. Configuramos o sistema dinamicamente quando estamos, por exemplo, em uma rede que não é a nossa e precisamos atribuir um certo endereço de IP e um gateway somente naquele momento.

Para atribuirmos na interface de rede `eth0` o IP: 192.168.1.75 dinamicamente, executamos o comando:

```
$ ifconfig eth0 192.168.1.75
```

Para verificarmos se o IP foi configurado corretamente na interface, usamos o comando:

```
$ ifconfig eth0
```

Neste caso, o resultado foi:

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.75  netmask 255.255.255.0  broadcast 192.1
68.1.255
        inet6 fe80::d250:99ff:feae:e646  prefixlen 64  scopeid 0x
20<link>
```

Observe que em `inet` temos o IP "192.168.1.75" e que em `netmask` temos o valor "255.255.255.0" que o Linux atribuiu automaticamente. Neste caso é um IP da classe C.

Atribuindo o DNS

Agora que já atribuímos um IP à nossa interface, vamos configurar nosso sistema para apontar para um servidor DNS. Neste caso, devemos editar um arquivo chamado `resolv.conf` , que fica dentro do diretório `/etc` .

Vamos abrir este arquivo usando o editor Vim:

```
$ vim /etc/resolv.conf
```

Dentro deste arquivo, você deve colocar o endereço do seu servidor DNS fornecido pelo administrador de sua rede ou pelo seu serviço de internet. Caso não saiba qual DNS utilizar, você pode usar o servidor DNS público do Google, que tem o endereço IP: 8.8.8.8.

O conteúdo do arquivo deve ser:

```
nameserver 8.8.8.8
```

Nesse exemplo, estou usando o DNS do Google, mas você pode alterar pelo DNS de sua rede.

Atribuindo o gateway

O último passo é atribuirmos o endereço do nosso gateway, pois é ele quem vai direcionar nosso acesso ao mundo externo com a internet. O gateway pode ser um servidor, normalmente um servidor de Firewall que protege a rede de ataques, um roteador, ou um modem adsl. O comando para atribuirmos um gateway é:

```
$ route add default gw 192.168.1.1
```

Observe que estamos definindo que o padrão (`default`) será o gateway (`gw`) de IP 192.168.1.1. Podemos visualizar o gateway configurado com o comando:

```
$ route -n
```

Neste caso, a saída é:

```
Tabela de Roteamento IP do Kernel
Destino         Roteador        MáscaraGen.    Opções Métrica Ref
```

```
   Uso Iface
0.0.0.0         192.168.1.1     0.0.0.0         UG    0      0
     0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0
     0 eth0
```

Observe na coluna `Roteador` o IP 192.168.1.1, que significa que a configuração foi um sucesso.

8.3 CONFIGURAÇÃO ESTÁTICA

Agora que já configuramos a nossa rede de forma dinâmica, vamos configurá-la de forma estática, ou seja, que se mantém mesmo após a reinicialização do sistema. Para isso, temos que editar o arquivo `/etc/network/interfaces` .

Abrimos o arquivo com o editor Vim:

```
$ vim /etc/network/interfaces
```

Dentro dele, vamos atribuir a seguinte configuração:

- IP: 192.168.1.40
- Netmask: 255.255.255
- Gateway: 192.168.1.1

A sintaxe do arquivo deve ser:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.1.40
netmask 255.255.255.0
gateway 192.168.1.1
```

Observe que atribuímos a configuração do endereço de

loopback (`lo`) e da interface ethernet (`eth0`).

Após salvar e fechar este arquivo, será necessário reiniciar o serviço de rede. Para isto, executamos o comando:

```
$ service networking restart
```

Mesmo reinicializando o computador, esta configuração se manterá.

8.4 CONFIGURANDO HOSTNAME

Dentro de uma rede de computadores, podemos usar nomes paras os hosts de modo a facilitar o acesso a estes equipamentos, semelhante ao que faz o servidor DNS que já abordamos.

Para criarmos estes nomes de hosts, devemos editar o arquivo: `/etc/hosts` .

A sintaxe do arquivo é bem simples:

IP	FQDN	Hostname	Alias
Endereço IP, Exemplo: 192.168.1.10	Full Qualified Domain Name - Nome da máquina com o nome do domínio, exemplo: grubelio.certificacoes.net.br	Nome da máquina, exemplo: grubelilo	Apelido (*opcional), exemplo: micro10

Criar estes nomes facilita a administração da rede, já que não precisamos decorar os números IPs. Veja um exemplo do arquivo `/etc/hosts` :

```
192.168.1.40        pc-juliano.certificacoes.net.br     pc-julian
o

192.168.1.41        note-keila.certificacoes.net.br    note-keila
```

```
192.168.1.42        servidor-samba.certificacoes.net.br    servid
or-samba
```

Alterando o nome da nossa máquina

Assim como procedemos com a configuração de rede estática e dinâmica, podemos definir o nome da máquina de forma estática (ativo no próximo boot) ou dinâmico (para uso somente nesta seção).

Para alterarmos o nome da máquina de modo dinâmico, usamos o comando:

```
hostname novo-nome
```

Para alterar o nome da máquina de forma estática, editamos o arquivo: `/etc/hostname` a sintaxe do arquivo é a mesma do arquivo `/etc/hosts` .

Quando usar estático e quando usar dinâmico?

Às vezes, eu visito empresas para realizar consultoria, levo meu notebook e conecto o mesmo no ponto de rede. Neste momento, faço uma configuração dinâmica, colocando o IP e máscara daquela rede, sendo assim, quando reiniciar meu equipamento ainda será válida minha configuração estática, que utilizo em casa na minha rede privativa.

Observe que a configuração dinâmica não vai estar disponível após a reinicialização do computador. Na maioria das vezes, usamos configuração estática nas máquinas, porque elas não vão ser levadas para outras redes.

8.5 CONCLUSÃO

Conhecer o básico sobre configurações de rede é fundamental para que você possa trabalhar como um administrador de sistemas Linux. Estude bem este capítulo, que é um dos mais importantes do livro. É aqui que você atravessa a linha entre ser um usuário de desktop e começa a ser um usuário de servidores.

Capítulo 9

INSTALAÇÃO, REMOÇÃO E ATUALIZAÇÃO DE PROGRAMAS

Instalar, remover e atualizar programas são tarefas cotidianas de um administrador de sistemas Linux. No Linux, os programas são fornecidos em forma de pacotes específicos para cada distribuição. Neste capítulo, aprenderemos sobre os pacotes oferecidos para a distribuição Debian e sistemas derivados, como o Linux Mint e Ubuntu.

Pacotes de software, como chamamos no mundo Linux, são um conjunto de binários (programas executáveis), bibliotecas, arquivos de controle e de configuração, que são facilmente instalados no sistema operacional. Podemos comparar os pacotes de software do Debian com os programas instaladores do Windows.

No Debian, os pacotes de software possuem a extensão `.deb`.

O gerenciador de pacotes

O gerenciador de pacotes é um software com a função de instalar, atualizar e remover programas no Linux. A grande

vantagem de se utilizar um gerenciador de pacotes é que ele tem uma missão extraordinária: ele resolve conflitos de dependências, que entenderemos a seguir.

Um pacote nem sempre depende apenas dele mesmo, ou seja, quando instalamos um programa, ele pode depender de bibliotecas de áudio, vídeo, imagens, funções e de até mesmo outro programa para funcionar. A este elo de programas, chamamos de dependências.

De um modo mais simplista, digamos que, para instalar um determinado programa chamado de A, precisamos de um software compilador, chamado B, e de uma biblioteca de sistema, chamada C. Quando vamos instalar o programa A, pelo gerenciador de pacotes, ele procura automaticamente na internet as dependências B e C em servidores de repositórios de software, e as instala junto com o pacote A.

No Debian 9 - Stretch, o gerenciador de pacotes é o **apt-get**.

9.1 GERENCIANDO PACOTES NO DEBIAN

O primeiro passo a se realizar em um sistema Debian é selecionar os repositórios que utilizaremos para buscar os pacotes que serão instalados e atualizados em nosso sistema. Para isto, editaremos o arquivo `/etc/apt/sources.list` .

O arquivo `/etc/apt/sources.list` contém os locais onde o `apt` encontrará os pacotes. Neste arquivo definimos a versão do nosso Debian, que pode ser:

- **Stable**: a versão atual do Debian, considerada estável.

Recomendo esta versão para equipamentos de produção. Seu tempo de vida costuma ser de 2 anos, com suporte total dos times de desenvolvimento e segurança do Debian.

- **Testing**: esta versão se torna a versão *stable*. Como o nome diz, ela está em uma fase de teste. Recomendo apenas para quem deseja conhecer as últimas aplicações que podem ou não estar depois disponíveis na versão *stable*.

- **Unstable**: esta é a versão em desenvolvimento, em fase experimental. É apenas para quem é desenvolvedor e deseja contribuir com o desenvolvimento do Debian.

No nosso caso, estamos usando a versão estável do Debian, versão 9, codinome **Stretch**.

Segue um exemplo do meu arquivo do `/etc/apt/sources.list`:

```
# deb cdrom:[Debian GNU/Linux 9.0.0 _Stretch_ - Official amd64 xf
ce-CD Binary-1 20170617-13:07]/ stretch main
# deb cdrom:[Debian GNU/Linux 9.0.0 _Stretch_ - Official amd64 xf
ce-CD Binary-1 20170617-13:07]/ stretch main

deb http://security.debian.org/debian-security stretch/updates ma
in
deb-src http://security.debian.org/debian-security stretch/update
s main

# stretch-updates, previously known as 'volatile'
# A network mirror was not selected during install.  The followin
g entries
# are provided as examples, but you should amend them as appropri
ate
# for your mirror of choice.
#
deb http://deb.debian.org/debian/ stretch-updates main
deb-src http://deb.debian.org/debian/ stretch-updates main
```

```
deb http://deb.debian.org/debian/ stretch main contrib non-free
```

Observe que eu comentei as linhas que se iniciam com `deb cdrom` , isto porque não desejo que meu sistema procure pacotes de software através do CD do Debian. Além disto, adicionei ao final do arquivo a linha:

```
deb http://deb.debian.org/debian/ stretch main contrib non-free
```

Esta linha vai permitir ao meu sistema procurar e instalar programas não livres, como o plugin do Flash player e diversos *codecs* multimídia. Após configurar seu arquivo, feche-o e execute o comando a seguir para atualizar os links dos repositórios em seu sistema:

```
# apt-get update
```

Esse comando sincroniza a lista de pacotes disponíveis nos servidores remotos (repositórios) para uma lista local. A lista local visa a acelerar as consultas e pesquisas de novos softwares que você desejar instalar futuramente.

Pesquisando um pacote

Para procurarmos por um pacote que desejamos instalar, podemos fazer uma busca através do comando:

```
# apt-cache search <argumento>
```

Em `<argumento>` , você pode definir o nome do programa que deseja procurar ou uma descrição. Exemplo:

```
# apt-cache search firefox
```

Ou:

```
# apt-cache search browser
```

Colocando `browser`, como no exemplo anterior, o resultado será uma lista com vários browsers disponíveis nos repositórios, assim você poderá escolher um para instalar. Sabendo agora o nome correto do pacote, você poderá obter uma maior descrição dele, inclusive, visualizando suas dependências, com o comando:

```
# apt-cache show <nome_do_pacote>
```

Instalando um pacote

Normalmente, não utilizamos interface gráfica em um servidor Linux, administrando todo o sistema pelo prompt de comando. Pode acontecer de, em algum momento em nossa administração de sistema, precisarmos de uma busca na Web. Neste caso, podemos usar o navegador em modo texto, `elinks`.

```
# apt-cache search elinks
elinks - avançado navegador web em modo texto
wv - programas para acessar documentos do Microsoft Word
circos-tools - plotter for visualizing data - helper utilities
elinks-data - advanced text-mode WWW browser - data files
elinks-doc - advanced text-mode WWW browser - documentation
gt5 - shell program to display visual disk usage with navigation
libhtml-formatexternal-perl - HTML to text formatting using exter
nal programs
```

Observe que, na saída do comando, além do elinks ele resultou em algumas outras opções. Como agora já sabemos que o elinks está nos repositórios, vamos obter mais informações sobre ele:

```
# apt-cache show elinks
Package: elinks
Version: 0.12~pre6-12
Installed-Size: 1587
Maintainer: Moritz Muehlenhoff <jmm@debian.org>
Architecture: amd64
Provides: www-browser
Depends: libbz2-1.0, libc6 (>= 2.15), libcomerr2 (>= 1.01), libex
```

```
pat1 (>= 2.0.1), libfsplib0 (>= 0.9), libgnutls30 (>= 3.5.3), lib
gpm2 (>= 1.20.4), libgssapi-krb5-2 (>= 1.14+dfsg), libidn11 (>= 1
.13), libk5crypto3 (>= 1.6.dfsg.2), libkrb5-3 (>= 1.6.dfsg.2), li
blua5.1-0, libperl5.24 (>= 5.24.0), libtre5, zlib1g (>= 1:1.1.4),
 elinks-data (= 0.12~pre6-12), debconf (>= 0.5) | debconf-2.0
Pre-Depends: dpkg (>= 1.17.14)
Suggests: elinks-doc
Description-pt_BR: avançado navegador web em modo texto
 ELinks é um programa repleto de recursos para navegar na web em
modo texto.
 É como se fosse os programas Lynx e Links aprimorados. Os recurs
os mais
 notáveis do ELinks são:
```

Aqui está apenas parte da descrição que foi obtida. Observe que o gerenciador de pacotes instalará diversas dependências para este pacote funcionar (*Depends*). Como desejamos ter este pacote, vamos realizar sua instalação:

```
# apt-get install elinks
```

Em alguns casos, você precisará confirmar a instalação com um `S` maiúsculo ou, caso seu sistema esteja em inglês, com um `Y`. Agora que o pacote foi instalado, você simplesmente o chama pelo seu nome:

```
# elinks
```

Quando você abre o elinks, a primeira tela é para você colocar a URL do site que deseja visualizar. A navegação dentro do site é realizada com as setas direcionais do teclado e `< enter >` para confirmar alguma ação ou acesso a um link. Para abrir novamente a busca de URL, utilize a tecla `< g >`, e para fechar o navegador, a tecla `< q >`.

Removendo um pacote

Para remover um pacote instalado, usamos o parâmetro `remove` . Exemplo:

```
# apt-get remove elinks
```

Removemos o navegador em modo texto elinks, que instalamos anteriormente. O único problema do parâmetro `remove` é que ele não resolve bem a remoção das dependências. Para solucionar isso, podemos usar o parâmetro `purge` . Veja no exemplo:

```
# apt-get purge elinks
```

Observe que, mesmo tendo removido o elinks com `remove` , ao usarmos o parâmetro `purge` ele encontrou resquícios (dependências) do programa que não foram removidas e as removeu.

Removendo pacotes que não são mais usados

Quando você instala um pacote, o `apt` busca nas fontes listadas em `/etc/apt/sources.list` os arquivos necessários para a instalação e os guarda em um repositório local em `/var/cache/apt/archives` , e então faz a instalação. Acontece que, com o passar do tempo, o repositório local pode ocupar muito espaço em disco, já que todos os pacotes de instalação ficam lá. Você pode visualizar os pacotes que lá estão com o comando:

```
# ls /var/cache/apt/archives
```

Para limpar este repositório local, usamos o comando:

```
# apt-get clean
```

Atualizando pacotes instalados

Para atualizar os pacotes já instalados, buscando no repositório a última versão, usamos o comando:

```
# apt-get upgrade
```

Já para atualizar a distribuição, usamos o comando:

```
# apt-get dist-upgrade
```

Compilando um programa

Alguns programas do Linux são disponibilizados através de código-fonte. Nestes casos, devemos realizar a sua compilação para que ele esteja funcional em nosso sistema. O primeiro passo e o mais importante é garantir que temos as ferramentas básicas para realizar a compilação destes programas. Para isto, vamos fazer a instalação dos pacotes:

```
# apt-get install build-essential bzip2 gzip unzip g++ wget curl
```

Agora que temos as ferramentas necessárias para compilar um programa, vamos fazer a instalação do programa `nmap` que utilizaremos nos próximos capítulos. O `nmap` é uma aplicação que visualiza quais portas de serviços do sistema estão abertas, sendo uma boa ferramenta para a segurança. O primeiro passo é fazer download do código-fonte do programa:

```
# wget https://nmap.org/dist/nmap-7.50.tar.bz2
```

O arquivo com o código-fonte está compactado no formato `bunzip2` , por isto, vamos descompactar usando o comando:

```
# tar xvjf nmap-7.50.tar.bz2
```

Agora, vamos acessar o diretório descompactado:

```
# cd nmap-7.50
```

Sempre que pegamos o código-fonte de um programa, ele virá com um aplicativo chamado `configure`, que vai executar uma verificação em seu sistema a fim de verificar se ele dispõe de todos os componentes básicos para uma compilação bem-sucedida.

Além disso, quando consultamos o `help` do `configure`, ele nos mostrará todas as funcionalidades que podemos compilar com o programa e todas as funcionalidades que podemos retirar dele. Uma vez que o processo de `configure` for encerrado com sucesso, ele gerará um arquivo chamado `Makefile` que contém instruções para a compilação do programa.

Visualizando o `help` do `configure` da aplicação `nmap`:

```
# ./configure --help
```

Ao executarmos esse comando, visualizamos algumas opções que podem ser utilizadas para a preparação da compilação do `nmap`. Como não estamos interessados na interface gráfica do `nmap`, podemos informar ao configure a opção `--without-zenmap`:

```
# ./configure --without-zenmap
```

Quando o `configure` terminar, podemos observar que ele gerou um arquivo chamado `Makefile`:

```
# cat Makefile
```

Agora sim, vamos compilar o programa:

```
# make
```

E por fim, procedemos com a instalação do programa compilado:

```
# make install
```

Para verificar se o `nmap` está funcionando corretamente, execute:

```
# nmap localhost
```

Esse comando vai retornar todas as portas abertas por serviços em seu sistema.

Removendo um programa compilado

Dentro do diretório da aplicação que você compilou, execute:

```
# make clean
# make uninstall
```

Instalando pacotes com DPKG

O DPKG é um programa que é a base do sistema de gerenciamento de pacotes para distribuições GNU/Linux baseadas em Debian. Criado por Ian Jackson em 1993, é usado para instalar, remover e fornecer informações sobre os pacotes `.deb` . O comando `dpkg` não resolve conflitos de dependências como o `apt-get` .

Você pode obter uma lista de parâmetros do `dpkg` executando o comando:

```
# dpkg --help
```

Antes de instalar um pacote `.deb` com o `dpkg` , você deve verificar algumas informações:

1. Este pacote `.deb` foi criado para o Debian? Alguns pacotes criados para Ubuntu/Mint não funcionam no Debian.

2. Este pacote é criado para minha arquitetura de sistema? Verifique se o pacote é 32 bits ou 64 bits.
3. Este pacote é compatível com minha versão do Debian? Pacotes antigos podem não funcionar em versões recentes do Debian.

Normalmente, estas informações estão disponíveis no site do programa que você deseja baixar/instalar. Caso não as encontre, tente algum contato com o desenvolvedor.

Para instalar o pacote, execute:

```
# dpkg -i nome_do_pacote.deb
```

Verifique se o pacote está instalado:

```
# dpkg -l nome_do_programa
```

Exemplo:

```
# dpkg -l elinks
```

Para verificar o status do pacote instalado:

```
# dpkg -s elinks
```

Remover um pacote instalado com `dpkg`:

```
# dpkg -r nome_do_pacote
```

O moderno sistema snap

Os pacotes `snap` foram originalmente desenvolvidos para o Ubuntu, no entanto, a Canonical (empresa que desenvolve o Ubuntu Linux) portou o sistema para outras distros, entre elas o Debian. A grande novidade do `snap` é que os seus pacotes já possuem todas as dependências, de modo que o gerenciador não

precisa instalar dependências adicionais.

Instalando o `snap` no Debian:

```
# apt-get install snapd
```

Com o pacote `snap` instalado, você pode procurar por aplicações disponíveis com o comando:

```
# snap find nome_do_snap
```

O `snap` não usa os repositórios do Debian, tendo seus próprios repositórios de software mantidos pela Canonical. Para instalar um pacote pelo `snap`, utilize:

```
# snap install nome_do_snap
```

Para remover um pacote `snap`, utilize:

```
# snap remove nome_do_snap
```

Para atualizar um pacote `snap`, utilize:

```
# snap refresh nome_do_snap
```

Para visualizar todos os `snaps` instalados em seu sistema, utilize:

```
# snap list
```

Para visualizar mais parâmetros do `snap`, utilize:

```
# snap -h
```

9.2 CONCLUSÃO

Instalar, remover e atualizar programas é uma tarefa cotidiana para qualquer administrador de sistemas. Procure sempre ter um

sistema virtualizado para testar novas aplicações antes de realizar sua instalação em servidores de produção e sempre utilize os repositórios oficiais do Debian. Alguns repositórios de aplicações de terceiros podem não ser confiáveis, por isto, evite utilizá-los em servidores.

Capítulo 10

SERVIDOR SSH

SSH ou *Secure Shell* é o protocolo que permite estabelecer um canal de comunicação seguro entre dois computadores. Para garantir esta segurança, SSH utiliza criptografia em sua comunicação. Basicamente, usamos SSH para nos conectarmos a computadores remotos e para executar comandos, entretanto, existe outra função no SSH que é muito utilizada pelos administradores de sistema: a transferência de arquivos.

Manutenção remota com segurança

Através do SSH você pode acessar um servidor que esteja na sala ao lado ou do outro lado do mundo, executar todos os comandos de administração de sistema como se estivesse na frente do equipamento e até mesmo transferir arquivos para este servidor ou deste servidor para seu computador. E o melhor, a sintaxe do SSH é muito simples e fácil de se utilizar.

Instalando o SSH

O primeiro passo é instalar o servidor SSH no seu Debian. Para isso, execute o comando:

```
# apt-get install openssh-server
```

O arquivo de configuração do `openssh-server` é `/etc/ssh/sshd_config`.

Dentro deste arquivo, as opções mais comumente utilizadas por um administrador de sistema são:

Comando do arquivo	Opções	Definição
`Port`	22	Altera a porta padrão do SSH que é de valor padrão 22.
`PermitRootLogin`	Yes/No	Habilita/Nega acesso do usuário `root` por SSH.
`Banner /etc/issue.net`	Yes/No	Habilita/Nega mensagem de login presente em `/etc/issue.net`
`LoginGraceTime`	15	Tempo para se logar no servidor em segundos.

Caso realize alguma alteração no arquivo de configuração do SSH, é necessário reinicializar o serviço:

```
# service sshd restart
```

Acessando uma máquina remota

A forma de utilização mais básica para se acessar uma conexão remota é através da sintaxe:

```
# ssh nome_do_usuario_remoto@ip_do_servidor_remoto
```

Exemplo:

```
# ssh juliano@192.168.1.10
```

Outra opção é se logar no servidor remoto com o mesmo nome de usuário com que você está logado, desde que este mesmo usuário exista remotamente.

```
# ssh ip_servidor
```

Se você está logado como `juliano`, por exemplo, e no servidor remoto tem um usuário chamado `juliano`, você pode usar a opção `ssh` sem o nome do usuário, porém, se este usuário não existe no servidor remoto, você deverá informar o nome do usuário.

Verificando a porta padrão do SSH

No capítulo anterior, aprendemos como instalar o `nmap` através do código-fonte. Caso você não tenha realizado esta tarefa, você pode instalá-lo pelos repositórios, usando:

```
# apt-get install nmap
```

Vamos utilizá-lo agora para verificar qual é a porta padrão do SSH:

```
# nmap localhost
```

No meu caso, o retorno do comando foi:

```
Starting Nmap 7.40 ( https://nmap.org ) at 2017-07-26 12:01 -03
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000013s latency).
Not shown: 996 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds
631/tcp open  ipp
```

Observe que a porta do SSH está como padrão (22). Caso você tenha outro número, quando desejar acessar esta máquina deverá realizar o comando:

```
# ssh -p Numero_da_porta nome_do_usuario@Ip_do_servidor
```

Exemplo:

```
# ssh -p 5498 juliano@192.168.1.10
```

Após executar esse comando com sucesso e digitar sua senha, você vai estar no prompt remoto, ou seja, no host com o IP que definiu, neste caso (192.168.1.10).

Descobrindo uma porta com o nmap

Podemos usar a opção `-p` do `nmap` que serve para passar um range de portas ou uma porta específica para ser escaneada, deste modo, temos uma probabilidade grande de descobrir qual porta o servidor remoto está usando para o SSH.

Exemplo do comando:

```
# nmap -p 0-65535 -sV ip_do_servidor
```

A opção `-sV` vai identificar qual serviço está sendo executado. No exemplo, estamos pesquisando em um IP portas abertas dentro do range de 0 a 65535.

Copiando arquivos para o servidor

Durante a administração de sistema, você pode precisar copiar arquivos de sua máquina local para o servidor remoto. Por exemplo, você criou em sua máquina local um script que realiza um procedimento de backup, agora deve mover para o servidor remoto. Se ele tivesse interface gráfica, seria fácil não é mesmo? Você poderia mandar por anexo em um e-mail, ou compartilhar nestas ferramentas de armazenamento na nuvem. Como um bom servidor Linux não tem interface gráfica, o melhor é usar o `scp` , assim você não precisa instalar um servidor FTP para copiar seus arquivos. A sintaxe para copiar um arquivo é:

```
# scp arquivo usuario@ip_de_destino:/destino
```

Exemplo:

```
# scp documento.txt juliano@192.168.1.10:/home/juliano/documentos
```

Para copiar um diretório, usamos:

```
# scp -r diretorio usuario@ip_de_destino:/destino
```

Exemplo:

```
# scp -r videos juliano@ip_de_destino:/home/juliano/
```

Copiando arquivos do servidor para máquina local

Já se você deseja copiar arquivos do servidor remoto para seu computador, execute:

```
# scp usuario@ip_servidor_remoto:/arquivo /destino
```

Exemplo:

```
# scp juliano@192.168.10.233:/home/juliano/beatles.mp3 /home/juli
ano/musicas/
```

Para copiar um diretório:

```
# scp -r juliano@192.168.10.233:/home/juliano/musica /home/julian
o/musicas/
```

Copiando arquivos com uma porta diferente

Diferente do SSH, em que usamos a sintaxe `-p` (minúsculo), para copiarmos arquivos usando uma porta diferente, usamos a sintaxe `-P` (maiúsculo). Exemplo:

```
# scp -P 6554 juliano@192.168.10.233:/home/juliano/beatles.mp3 /h
ome/juliano/musicas/
```

10.1 SSH COM CHAVES ASSIMÉTRICAS

Chaves assimétricas são uma forma de aumentar a segurança de uma comunicação SSH. Basicamente, consiste na geração de dois arquivos que contêm sequências aleatórias e que só têm funcionalidade se os dois trabalharem em conjunto. Ou seja, quando criamos um par de chaves, serão criadas uma chave pública e uma chave privada. A chave privada é sua e absolutamente ninguém deve ter acesso a ela; a sua chave pública você coloca no servidor remoto, como veremos mais à frente.

Quando você tentar estabelecer uma conexão, ela só será possível se a chave privada se encaixar com pública. Com este sistema, existe uma única chave privada que se encaixa em uma única chave pública. Desta forma, apenas quem possuir a chave privada poderá estabelecer uma conexão utilizando a respectiva chave pública.

Você pode criar um par de chaves assimétricas para aumentar o nível de segurança, usando uma senha, ou facilitar a execução de scripts remotos, não definindo uma senha.

De um modo mais simples, você cria uma chave e uma fechadura. Entrega a chave para o servidor remoto e ele somente poderá acessar o sistema se a sua chave (pública) confirmar com a fechadura (chave privada na máquina local).

Criando as chaves assimétricas

Execute na sua máquina local o comando:

```
# ssh-keygen -t rsa
```

Por padrão, será criado o diretório `/root/.ssh/` e dentro dele você terá a chave pública e privada. Durante a execução do comando anterior, você será questionado a criar uma frase de segurança. Caso não coloque senha alguma, você poderá realizar acesso ao servidor remoto sem precisar colocar senha e de um modo seguro. Se optar por colocar senha, você tornará a conexão ainda mais segura, porque, além da senha, a chave pública do servidor deverá ser compatível com a chave privada da sua máquina local.

Para copiar a chave pública ao servidor, utilize o comando:

```
 # ssh-copy-id -i ~/.ssh/id_rsa.pub usuario@ip_do_servidor_remoto
```

Exemplo:

```
# ssh-copy-id -i ~/.ssh/id_rsa.pub root@192.168.1.10
```

Caso você não tenha colocado nenhuma senha, seu acesso ao servidor será automático, quando executar:

```
# ssh root@192.168.1.10
```

Ou seja, ele não vai pedir senha, porque tanto o seu computador quanto este remoto (192.168.1.10) possuem chaves assimétricas.

10.2 CONCLUSÃO

Obter acesso remoto a um servidor ou estação Linux é necessário para quebrar barreiras geográficas e reduzir custos com deslocamento. Em muitas situações, você poderá atuar em *home office* ou de qualquer lugar, bastando apenas ter uma conexão com a internet e uma aplicação que funcione por SSH. No caso do

Microsoft Windows é possível instalar um cliente SSH chamado Putty (http://www.putty.org/).

Capítulo 11

PARTICIONAMENTO DE DISCO

Todo administrador de sistemas deve compreender o conceito de particionamento de disco e realizar diversos tipos de particionamento com diferentes sistemas de arquivos. Uma partição é um espaço em disco que se destina a receber um sistema de arquivos ou, em um caso particular que veremos adiante, outras partições.

Um sistema de arquivos é uma forma de organização de dados em um meio de armazenamento (HD, pendrive, DVD, Blu-Ray) em massa. Em uma analogia, o sistema operacional é o bibliotecário e o disco é a biblioteca. O sistema de arquivos é a forma de se categorizar os livros ou dados. É importante saber que cada sistema de arquivos tem seu próprio meio de categorizar os dados, sendo assim, não é possível ter dois ou mais tipos de sistemas de arquivos (formatos) na mesma partição.

O MBR (*Master Boot Record*) é um arquivo de dados interligado com a BIOS (*Basic Input/Output System*), que é uma *firmware* usada para realizar a inicialização do hardware, cuja importância é o reconhecimento do sistema de arquivos e o boot (inicialização) do sistema operacional. Os tipos mais comuns de

sistema de arquivos são:

Apple Macintosh (Mac OS)

1. HFS
2. HFS+

UNIX (FreeBSD, OpenBSD, Linux, Solaris, Android etc.)

1. Ext4
2. Ext3
3. SWAP
4. ReiserFS
5. JFS
6. XFS
7. ZFS
8. HPFS
9. UFS

MS-DOS/Microsoft Windows

1. FAT16
2. FAT32
3. NTFS

11.1 TIPOS DE PARTIÇÕES

Existem três tipos possíveis de partições: primária, estendida e lógica. Em um disco padrão, você pode criar 16 partições. Sendo elas:

1. **3 partições primárias**
2. **1 partição estendida, que é uma partição primária**

3. **12 partições lógicas**

Partições primárias

A partição primária contém um sistema de arquivos. Em um disco, deve haver no mínimo uma e no máximo quatro partições primárias. Se existirem quatro partições primárias, nenhuma outra partição poderá ser criada neste disco. As partições primárias no Linux são nomeadas da seguinte forma:

```
/dev/sda1
/dev/sda2
/dev/sda3
/dev/sda4
```

Partição estendida

Só pode haver uma partição estendida em cada disco. Uma partição estendida é um tipo especial de partição primária que não pode conter um sistema de arquivos, ou seja, você não poderá armazenar nenhum dado nesta partição. Ela é usada para permitir a criação de outras partições, que chamamos de lógicas. Não podemos armazenar arquivos na partição estendida, por isto, caso utilize as 16 partições possíveis em um disco, efetivamente só poderá armazenar arquivos em 15 delas.

Por exemplo, digamos que você precise criar 6 partições em um disco. No Linux vai ficar com as nomenclaturas:

```
/dev/sda1 (primária)
/dev/sda4 (estendida)
/dev/sda5 (lógica)
/dev/sda6 (lógica)
/dev/sda7 (lógica)
/dev/sda8 (lógica)
```

Observe no exemplo que `/dev/sda2` e `/dev/sda3` não aparecem, isto porque quando você cria uma partição estendida, não é mais possível criar partições primárias. Como as partições primárias vão de `/dev/sda1` a `/dev/sda4` , o particionador automaticamente cria como `/dev/sda4` a primeira partição estendida, que é a que vai armazenar outras partições, as lógicas.

Partições lógicas

As partições lógicas são armazenadas dentro de uma partição estendida. Caso você remova a partição estendida, automaticamente, remove as partições lógicas que estão dentro dela.

Vamos fazer uma analogia das partições com uma gaveta de cozinha. Uma gaveta sem separador é como uma partição primária: suas colheres, garfos e facas ficam todos misturados. Já uma gaveta em que você coloca separadores para garfos, colheres e facas é como uma partição estendida e cada espaço do separador é uma partição lógica. Você não coloca mais os talheres direto na gaveta e sim nas repartições. Porém, se você remover esta gaveta, também vai remover todos os talheres que estão nestes separadores (isto é, nas partições lógicas).

11.2 O PARTICIONADOR FDISK

Apesar de sua interface pouco amigável, o FDISK é uma aplicação de particionamento completa. Trata-se de um utilitário presente em diversos sistemas operacionais que realiza particionamento de discos rígidos. Há versões do FDISK para Linux, DOS, Windows, FreeDOS e OS/2. No caso do DOS, ele

manipula apenas partições FAT. Sua sintaxe é:

```
# fdisk [ dispositivo ]
```

Exemplo:

```
# fdisk /dev/sdb
```

Quando fazemos a chamada do programa, temos a seguinte tela inicial:

```
The number of cylinders for this disk is set to 14593.
There is nothing wrong with that , but this is larger than 1024 ,
and could in certain setups cause problems with :
1) software that runs at boot time ( e . g . , old versions of LI
LO )
2) booting and partitioning software from other OSs
Command ( m for help ) :
```

Pressionando a tecla `< m >` obtemos um `help` :

```
Command ( m for help ) : m
Command action
a toggle a bootable flag
b edit bsd disklabel
c toggle the dos compatibility flag
d delete a partition
l list known partition types
m print this menu
n add a new partition
o create a new empty DOS partition table
p print the partition table
q quit without saving changes
s create a new empty Sun disklabel
t change a partition ' s system id
u change display / entry units
v verify the partition table
w write table to disk and exit
```

Para criarmos uma nova partição devemos, antes, ver se temos espaço disponível para isso, ou seja, precisamos imprimir a tabela de partições utilizando a letra `< p >` . Se houver espaço disponível

para a criação de uma nova partição, basta pressionar a letra `< n >` e informar o tipo de partição (primária ou estendida) e seu tamanho.

Exemplo de criação de uma partição:

```
Comando ( m para ajuda ) : n
Comando - a ç ã o
l lógica (5 ou superior )
p partição primária (1 -4)
```

Se observar na saída desse comando, você deve definir o tipo de partição, entre lógica e/ou partição primária. Como é a nossa primeira partição, podemos utilizá-la como primária, usando a tecla `< 1 >`. Você poderia usar como lógica, mas neste caso ela começaria com o número 5, por exemplo, `/dev/sda5` e você teria 4 partições a menos para utilizar futuramente.

```
1
```

Agora devo escolher o local de início da partição (usar o padrão) apenas aperte `< enter >`:

```
Primeiro cilindro (1 -1044 , padrão 1044) : 1
Usando valor padrão 1
```

Para o tamanho da partição, utilize esta sintaxe `+tamanho`. Exemplo:

```
+10G
```

No exemplo, estou criando uma partição de 10 gigabytes. Não esquecer do `G` (Gigabyte), `M` (Megabyte) ou `K` (Kilobyte).

11.3 O PARTICIONADOR CFDISK

A ferramenta `cfdisk` é mais amigável que o `fdisk`. Quando você abre o `cfdisk` consegue visualizar todas as partições em colunas, diferente do `fdisk`, que não as exibe. A navegação pelo `cfdisk` também é mais simples, usando-se as setas direcionais e `< enter >`, enquanto no `fdisk` tudo é realizado com teclas específicas.

Para acessar o particionador `cfisk`, execute o comando:

```
# cfdisk [ dispositivo ]
```

Exemplo:

```
# cfdisk /dev/sdb
```

Uma vez executado este comando, a tela do `cfdisk` se abrirá. A sua tela é muito intuitiva: utilizando as setas para cima e para baixo, você navega pela listagem das partições; utilizando as setas para esquerda e direita, você navega pelo menu na parte inferior da tela.

Para criar uma nova partição, selecione a opção `new` no menu inferior. Escolha o tipo de partição (primária ou estendida) e, em seguida, defina o espaço da partição. Lembre-se, até 4 partições, podem ser todas primárias. Caso deseje mais do que 4 partições, deverá criar 3 primárias, 1 estendida e as outras, lógicas.

```
10G
```

Note que, diferente do `fdisk`, no `cfdisk` não colocamos o sinal de adição (`+`) à frente do valor do tamanho da partição.

Após realizar estas tarefas, escolha no menu inferior a opção `Write` para salvar as mudanças. Uma pergunta vai aparecer para que você confirme as alterações. Responda **yes** ou **sim**, se estiver

em inglês ou português.

Após salvar feche o `cfdisk`, reinicie e visualize a partição com o comando:

```
# cat /proc/partitions
```

Com a partição criada é necessário aplicar um sistema de arquivos.

11.4 APLICANDO UM SISTEMA DE ARQUIVOS

Para que possamos gravar informações na partição que criamos, é necessário dar forma a ela usando um sistema de arquivos ou *filesystem*. Para aplicar o sistema de arquivos `ext4` padrão do Debian Linux, usamos o comando:

```
# mkfs.ext4 /dev/sdb1
```

Troque "sdb1" para o nome e número de sua partição.

Imagine que você comprou um HD externo USB para armazenar arquivos de backup. Mas este HD também será usado por outra pessoa que utiliza o sistema operacional Microsoft Windows. O Windows não será capaz de ler estes arquivos se o formato estiver como EXT4, padrão do Debian Linux. Para ter esta compatibilidade, é necessário usar o formato padrão do Windows, NTFS. O Debian é capaz de ler e escrever neste formato, desde que você instale a aplicação `ntfs-3g`, como veremos a seguir:

```
# apt-get install ntfs-3g
```

Agora podemos aplicar o `filesystem` com o comando:

```
# mkfs.ntfs /dev/sdb1
```

Para visualizar o consumo das partições, utilize o comando:

```
df -h
```

Para visualizar o consumo de um arquivo ou diretório em específico, utilize o comando:

```
du -h /tmp
```

Nesse exemplo, estou verificando o tamanho do diretório `/tmp`.

```
du -h /home/juliano/filme.mp4
```

Agora verifiquei o tamanho do arquivo `filme.mp4`.

Ponto de montagem

Para acessar a partição que criamos e aplicarmos um filesystem, é necessário um ponto de montagem, que nada mais é que um diretório que serve como um link para o dispositivo físico. No Linux, o diretório `/mnt` (*mount*) pode ser usado como um ponto de montagem, mas saiba que você pode montar este acesso usando qualquer diretório do sistema.

O primeiro passo é criarmos o nosso ponto de montagem:

```
# mkdir /mnt/backup
```

Aqui, criei um diretório chamado `backup` que servirá para acessarmos nossa partição:

```
# mount -t ext4 /dev/sdb1 /mnt/backup
```

O comando `mount` é utilizado para criar o link entre o ponto de montagem e o dispositivo físico (partição). A opção `-t` é usada

para você definir o tipo de filesystem. Caso esta partição estivesse usando o filesystem NTFS, usaríamos:

```
# mount -t ntfs /dev/sdb1 /mnt/backup
```

Após executar o comando, acesse os arquivos da partição no endereço: `/mnt/backup` .

Para desmontar esta partição, ou seja, parar sua utilização no sistema, execute:

```
# umount /dev/sdb1
```

Lembre-se de que não é possível montar uma partição estendida, ela é apenas um espaço para as partições lógicas.

Montagem automática das partições no boot

Agora que aprendemos a montar dispositivos, há um arquivo que facilitará nossa vida: `/etc/fstab` . Nele, devem estar as informações a respeito da montagem de todas as partições do sistema. Para tornar o acesso à partição `/dev/sdb1` automático no boot, dentro do arquivo `/etc/fstab` adicione a seguinte linha:

```
/dev/sdb1 /mnt/backup ext4 defaults,user 0 0
```

Vamos entender melhor cada parte desta linha:

Dispositivo	Ponto de montagem	Filesystem	Configurações	Verificação
`/dev/sdb1`	`/mnt/backup`	ext4	defaults, user	0 0

- **Dispositivo:** caminho da partição local ou endereço de rede.

- **Ponto de montagem:** caminho do diretório criado para acessar a partição.
- **Filesystem:** tipo de sistema de arquivos.
- **Configurações:** opções de montagem; `defaults` é um conjunto de regras para o acesso a partição que inclui leitura e escrita; a opção `user` determina que todos os usuários podem utilizar a partição.
- **Verificação**: o primeiro valor pode ser 0 e 1. Havendo um sistema de backup (*dump*) configurado, deverá ser efetuado o backup se ativo (1) ou ignorado (0). O segundo valor aceita valores de 0 a 2. Ele serve para realizar a checagem da integridade do sistema de arquivos. O valor 1 deve ser especificado para o `/` (raiz), e o valor 2 deve ser especificado para qualquer outra partição.

11.5 MEMÓRIA SWAP

Este tipo de filesystem é utilizado para fornecer ao sistema uma memória "virtual" que é adicionada a memória RAM instalada no sistema. Esta memória virtual é acionada quando sua memória RAM está chegando à sua totalidade de uso, para que o sistema não trave.

Como somente os dados da memória RAM são processados diretamente pelo processador, por ser mais rápida, quando você executar uma aplicação e a memória RAM começar a encher, o Linux moverá automaticamente os dados que estão sendo utilizados para a partição SWAP e liberará mais memória RAM para continuar carregando a aplicação.

Quando os dados movidos para a partição SWAP são

solicitados, o GNU/Linux troca estes dados com os que estão na memória RAM. Por este motivo, a partição SWAP também é chamada de área de troca.

Para aplicar o filesystem SWAP a uma partição, usamos o comando:

```
# mkswap /dev/sdb2
```

Para ativar esta partição, usamos:

```
# swapon /dev/sdb2
```

Para visualizar as partições `swaps` ativas, usamos o comando:

```
# cat /proc/swaps
```

Ou:

```
# swapon -s
```

Para desabilitar a partição SWAP, usamos:

```
# swapoff /dev/sdb2
```

11.6 CONCLUSÃO

Dificilmente você vai administrar um servidor Linux sem precisar, em algum momento, particionar ou acessar partições. Este é um trabalho até que rotineiro de um *sysadmin*. Use e abuse do Virtualbox para criar discos virtuais, criando e removendo partições. A vantagem de se usar virtualização é que em momento algum você terá o risco de apagar dados importantes do seu disco.

Capítulo 12

QUOTAS DE DISCO

O uso de um **sistema de quotas** é um assunto tão importante para o administrador de sistemas quanto o particionamento do disco rígido. Com este sistema, podemos limitar o tamanho do espaço de uma partição que um usuário poderá utilizar.

Imagine que você tem um HD de 1TB e 10 usuários. Um dos usuários utiliza 900GB deixando apenas 100GB para os outros 9 usuários. Isso é meio injusto, não é? Dentro de uma empresa, muitas vezes limitamos os recursos de acordo com o perfil do profissional. Um diretor, por exemplo, pode ter 200GB de espaço, enquanto um estagiário deverá ter apenas 10GB de espaço.

Tanto o sistema de arquivos quanto o kernel em uso devem possuir suporte a quotas. Uma vez que o sistema de arquivos suporta quotas, devemos adicionar os parâmetros de montagem `usrquota` e `grpquota` ao sistema de arquivos. Isso é feito através da edição do arquivo `/etc/fstab`. O `ext4`, que é o sistema de arquivos padrão do Debian 9, possui suporte a quotas.

O primeiro passo é instalar o pacote de quota, por meio do comando:

```
# apt-get install quota
```

Eu adicionei um novo HD em meu sistema, que vai suportar o

sistema de quotas. No seu caso, você poderá criar um HD virtual pelo próprio Virtualbox. Após o boot, este HD ficou reconhecido como `/dev/sdb`.

Depois de criar uma partição e aplicar o sistema de arquivos `ext4`, vou configurar o sistema de quotas nele, por meio do `/etc/fstab`, colocando as linhas a seguir no arquivo:

```
/dev/sdb1    /mnt/backup ext4 defaults,user,usrquota,grpquota 0 2
```

Salve e feche o arquivo.

Observe que o ponto de montagem será `/mnt/backup`, então vamos criar este diretório:

```
# mkdir /mnt/backup
```

Agora vamos reiniciar o sistema para as opções `usrquota` e `grpquota` estarem disponíveis.

Quando o sistema iniciar, você perceberá que na partição tem dois arquivos: `aquota.group` e `aquota.user`:

```
# ls /mnt/backup
```

Isso significa que a partição está pronta para a criação das quotas dos usuários.

Definindo a quota do usuário

Vamos supor que ficou definido pela diretoria da empresa que o usuário `juliano` deverá ter apenas 50MB com um limite máximo de 60MB de espaço. Para definir esta quota do usuário `juliano`, digitamos:

```
# edquota -u juliano
```

Dentro do `edquota`, faremos as configurações para que a quota do usuário `juliano` seja de 50MB e limite máximo de 60MB:

```
Disk quotas for user juliano ( uid 1001) :
Filesystem    blocks    soft    hard    inodes    soft    hard
/dev/sdb1    0        50000    60000    0        0        0
```

Só alteramos a coluna `soft` e `hard` colocando os valores em KB. Observe que após a coluna `inodes` temos outras duas colunas chamadas `soft` e `hard`. Neste caso, elas são usadas para limitar a quantidade de arquivos que os usuários podem armazenar, o que não faz muito sentido, porque você poderia colocar 10 arquivos, sendo que eles podem ser imagens de DVD com 4GB.

Sendo assim, o melhor é sempre limitar por tamanho como fizemos. Nunca altere o valor da coluna `blocks`, que informa o tamanho real em KB utilizado, e `inodes`, que mostra a quantidade de arquivos utilizada.

Agora salve o arquivo e feche-o.

Verificando as quotas atribuídas

Para verificar se a quota do usuário `juliano` foi aplicada com sucesso, execute:

```
# repquota -v -a
```

O limite máximo, ou seja, os 10MB acima dos 50MB permitidos que definimos, possui uma quantidade de dias para uso, que é chamada de **grace period**. Para mudar este tempo, execute:

```
# edquota -t
```

Para alterar o tamanho das quotas atribuídas, é só executar novamente o comando:

```
# edquota -u juliano
```

Definindo quota por grupo

O procedimento é praticamente o mesmo do que definir a quota para usuário. O que muda é o opção `-g`, de *group*. Digamos, que temos um grupo chamado `financeiro` e queremos definir uma quota de 500MB para todos os seus usuários. O comando seria:

```
# edquota -g financeiro
```

Edite:

```
Disk quotas for group financeiro ( gid 100) :
Filesystem    blocks    soft    hard    inodes    soft    hard
/dev/sdb1    0        500000    600000    0        0        0
```

Para verificar se a quota foi aplicada:

```
# edquota -vag
```

Replicando quota a novos usuários

Edite o arquivo `/etc/adduser.conf` e adicione um usuário que já tenha uma quota definida na opção `QUOTAUSER`.

Por exemplo: o usuário `juliano` já tem a cota de 50MB e vamos definir que os próximos usuários criados também vão receber esta mesma cota. Para isso, no arquivo, deixe:

```
QUOTAUSER="juliano"
```

Quando você criar um novo usuário, ele já virá com esta cota definida.

12.1 CONCLUSÃO

Particionar disco e gerenciar as quotas é um assunto muito importante dentro da administração de sistemas Linux. Sabendo criar usuários, particionar discos e gerenciar quotas, chegamos a 50% da sua formação como um *sysadmin*. Quotas também é um assunto da certificação LTC (*Linux True Certificate*). Vamos fazer aquela pausa para o café e acelerar para o próximo capítulo!

Capítulo 13

ARQUITETURA DO KERNEL LINUX

Ao instalar o Debian ou outra distribuição Linux, estamos utilizando o kernel que foi compilado pelos desenvolvedores da distribuição. Este kernel padrão deve ser capaz de rodar em praticamente qualquer PC e dar suporte a diversos tipos de recursos que o usuário pretenda utilizar, como sistemas de arquivos, suporte a USB, a diferentes tipos de protocolos de rede, entre outros.

O desenvolvedor compila um kernel básico, com poucas funcionalidades, e cria pedaços de código com funções específicas que podem ser adicionados a este kernel. Estes pedaços de código são chamados de módulos.

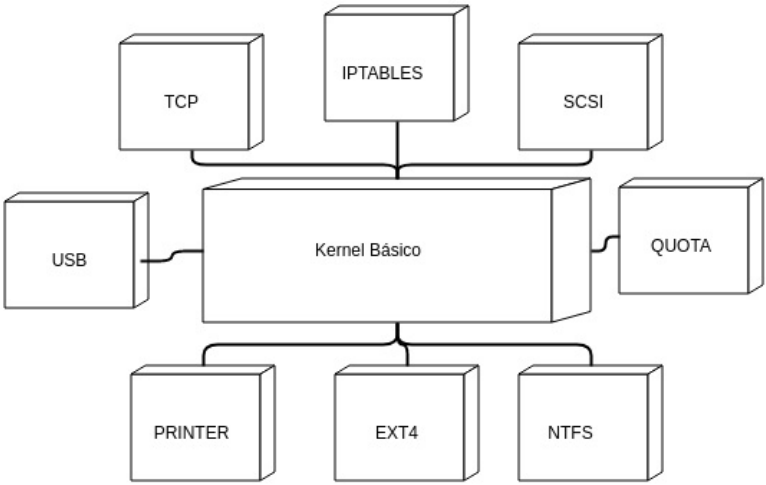


Figura 13.1: Módulos do kernel

Quando o sistema é carregado, este kernel básico é colocado na memória e passa a controlar a máquina. Digamos que você possui uma partição com o sistema de arquivos NTFS que não está no kernel básico padrão. Durante a inicialização do sistema, o kernel básico vai procurar se o módulo que dá suporte para este tipo de sistema de arquivos está disponível. Se este módulo for encontrado, ele será carregado, expandindo as funcionalidades do kernel.

13.1 TRABALHANDO COM MÓDULOS

O desenvolvimento de uma nova funcionalidade para o kernel Linux pode ser implementado diretamente no kernel ou compilado como um módulo. De modo geral, a escolha tende a ser a compilação como módulo. Isso significa que o kernel só vai

ativar o módulo caso seja necessário. Esta opção otimiza o sistema, tornando o kernel do Linux muito dinâmico.

Vamos a um exemplo prático: digamos que você esteja usando uma impressora USB. Neste momento, o módulo para ela funcionar está ativo no sistema. Depois de um tempo, você para de usar esta impressora por qualquer motivo, e o módulo de suporte a impressora não é mais necessário e você pode desabilitá-lo.

Ao executar o comando `lsmod` , você pode visualizar quais módulos estão carregados atualmente na memória:

```
# lsmod
Module                  Size  Used by
fuse                   98304  3
pci_stub               16384  1
vboxpci                24576  0
vboxnetadp             28672  0
vboxnetflt             28672  0
vboxdrv               454656  3 vboxnetadp,vboxnetflt,vboxpci
xfrm_user              36864  2
xfrm4_tunnel           16384  0
tunnel4                16384  1 xfrm4_tunnel
ipcomp                 16384  0
xfrm_ipcomp            16384  1 ipcomp
esp4                   20480  0
ah4                    20480  0
af_key                 36864  0
xfrm_algo              16384  5 xfrm_user,esp4,ah4,af_key,xfrm_ip
comp
snd_usb_audio         180224  2
uvcvideo               90112  0
snd_usbmidi_lib        28672  1 snd_usb_audio
snd_rawmidi            32768  1 snd_usbmidi_lib
videobuf2_vmalloc      16384  1 uvcvideo
videobuf2_memops       16384  1 videobuf2_vmalloc
videobuf2_v4l2         24576  1 uvcvideo
videobuf2_core         40960  2 uvcvideo,videobuf2_v4l2
wl                   6365184  0
videodev              176128  3 uvcvideo,videobuf2_core,videobuf2
_v4l2
```

```
media                  40960  2 uvcvideo,videodev
snd_seq_device         16384  1 snd_rawmidi
arc4                   16384  2
```

Cortei a saída inteira porque o meu kernel tem muitos módulos ativos. Estes módulos ativos são diferentes de máquina para máquina, já que possuímos hardware e configurações diferentes. Durante a instalação do Debian, o sistema verifica seu equipamento e define quais módulos vão estar ou não ativos no seu sistema.

Os módulos possuem dependências. Você pode saber a quantidade de dependências por módulos na coluna `used` na saída do comando `lsmod` . A coluna `size` mostra o tamanho do módulo em bytes. Um módulo só pode ser removido se o número de dependências for zero. Caso um módulo tenha dependências, ele não poderá ser removido.

Os módulos são carregados através do comando `insmod` . Se o módulo B depende do A, este deve ser carregado antes. Digamos que na coluna `used` do módulo A agora esteja mostrando 1, referente ao módulo B que você carregou. Não será possível remover o módulo A. Neste caso, você deve remover sua dependência B primeiro e só depois removê-lo.

Quais módulos estão disponíveis?

Para saber quais módulos estão disponíveis no seu sistema, ou seja, que não estão carregados na memória, acesse o diretório: `/lib/modules/versao-do-kernel/kernel` . No meu caso:

```
# cd /lib/modules/4.9.0-debian-amd64/kernel/
```

O meu kernel chama-se `4.9.0-debian-amd64` . O seu

provavelmente terá um nome diferente, já que este kernel que estou utilizando não é o padrão do Debian. Dentro deste diretório, você vai encontrar alguns subdiretórios como:

```
arch  crypto  drivers  fs  lib  mm  net  security  sound  virt
```

Dentro de cada um deles você encontra os módulos. Observe o diretório `fs` (filesystem). É ele que contém módulos sobre sistema de arquivos e dentro dele você pode visualizar todos os diretórios dos módulos disponíveis de `fs` :

```
# cd /lib/modules/4.9.0-debian-amd64/kernel/fs
# ls
9p       befs            ceph      dlm       ext4      fuse     j
bd2        minix       nilfs2  overlayfs  reiserfs  udf
adfs     bfs             cifs      ecryptfs  f2fs      gfs2     j
ffs2       ncpfs       nls     pstore     romfs     ufs
affs     binfmt_misc.ko  coda      efivarfs  fat       hfs      j
fs         nfs         ntfs    qnx4       squashfs  xfs
afs      btrfs           configfs  efs       freevxfs  hfsplus  l
ockd       nfs_common  ocfs2   qnx6       sysv
autofs4  cachefiles      crypto    exofs     fscache   isofs    m
bcache.ko  nfsd        omfs    quota      ubifs
```

No meu caso, tenho diversos módulos de sistemas de arquivo. Dentro do diretório do módulo de sua escolha, você obtém o nome do módulo. Exemplo:

```
# cd /lib/modules/4.9.0-debian-amd64/kernel/fs/fat
# ls
# fat.ko  msdos.ko  vfat.ko
```

Obseve que, dentro do diretório `/lib/modules/4.9.0-debian-amd64/kernel/fs/fat/` , obtive o nome dos módulos disponíveis `vfat` , `msdos` e `fat` .

A seguir, vamos observar quais são os procedimentos para carregar o módulo `vfat` , que dá suporte ao sistema de arquivos

FAT32, muito comum em pendrivers formatados no Microsoft Windows, e depois veremos como é o procedimento para sua remoção.

O primeiro passo é verificar se o módulo em questão já não está ativo no sistema. Para isso, executamos:

```
# lsmod | grep vfat
```

Caso não obtenha nenhum retorno, significa que o seu módulo não está ativo. Para ativarmos este módulo usamos a sintaxe:

```
# modprobe nome_do_modulo
```

Neste caso:

```
#modprobe vfat
```

Você pode visualizar o módulo ativo no sistema com o comando `lsmod` . O resultado deverá ser algo como:

```
vfat                   20480  0
fat                    69632  1 vfat
```

Para saber informações sobre um módulo e ver suas dependências, utilize o comando `modinfo` . Exemplo:

```
# modinfo vfat
filename:       /lib/modules/4.9.0-deepin9-amd64/kernel/fs/fat/vf
at.ko
author:         Gordon Chaffee
description:    VFAT filesystem support
license:        GPL
alias:          fs-vfat
depends:        fat
intree:         Y
vermagic:       4.9.0-deepin9-amd64 SMP preempt mod_unload modver
sions
```

Observe na saída que a dependência para o `vfat` chama-se

`fat`. Ao ativar o `vfat`, vamos obter novamente informação sobre ele:

```
# lsmod | grep vfat
vfat                   20480  0
fat                    69632  1 vfat
```

Veja que `fat` tem o número 1 na saída da coluna `used` Tentaremos removê-lo:

```
# rmmod fat
rmmod: ERROR: Module fat is in use by: vfat
```

Foi impossível remover, já que ele está sendo usado por `vfat`. Então vamos remover o `vfat` primeiro:

```
# rmmod vfat
```

O modo `vfat` foi removido com sucesso, mas ainda temos a dependência `fat` que precisa ser removida, logo:

```
# rmmod fat
```

A melhor forma de se remover um módulo é usando o modo automático, porque ele remove também todas as dependências. Para isso, utilizamos o comando:

```
# modprobe -r vfat
```

O bom neste caso é que, além de remover o `vfat`, ele remove o `fat`, sua dependência.

13.2 COMPILANDO UM KERNEL

Às vezes, algumas funcionalidades de que necessitamos só estão disponíveis em uma versão mais nova do kernel. Por exemplo, recentemente eu comprei um controle de Xbox para

jogar alguns games no Linux. Mas descobri que o módulo para ativar este joystick só estava disponível na versão mais nova do kernel. Então, fui obrigado a compilar um kernel novo e não mais utilizar o padrão, oferecido pela distribuição.

No site oficial do kernel (http://www.kernel.org), na frente de cada versão do kernel tem um link chamado *changelog* com informações sobre todas as novas funcionalidades daquele kernel.

O processo a seguir de compilação serve para qualquer versão de kernel que desejar instalar, bastando mudar os números que referenciam a versão. O primeiro passo que você deve realizar é instalar as dependências, que são obrigatórias, ou seja, não é possível instalar o kernel novo sem realizar este passo:

```
# apt-get install build-essential libncurses5-dev
```

O segundo passo é fazer download do kernel no site oficial: http://www.kernel.org/.

A versão escolhida foi a última estável no momento da edição do livro: 4.12, no link: https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.12.7.tar.xz/.

O terceiro passo é descompactar o kernel que baixei dentro do diretório `/usr/src` :

```
# tar -xvf linux-4.12.7.tar.xz -C /usr/src
```

Após descompactar, o quarto passo foi acessar o diretório e criar um link (atalho) para o diretório descompactado `linux-4.12.7` :

```
# cd /usr/src
# ln -s linux-4.12.7 linux
```

Observe que criei um atalho chamado `linux` que aponta para `linux-4.12.7` . Agora vamos acessar o diretório `linux` e executar o comando para acessar o menu de configuração do kernel `menuconfig` :

```
# cd linux
# make menuconfig
```

Nesta janela de opções, você pode ativar novas funcionalidades para o kernel. Por padrão, vamos apenas salvar (`save`). Isto vai gerar um arquivo `.config` necessário para o resto do processo. Nesta quinta etapa, devemos gerar a imagem do kernel propriamente dita, com o comando:

```
# make bzImage
```

Este processo demora um bom tempo, tenha paciência. No meu caso, demorou três horas. O sexto passo foi compilar os módulos, com o comando:

```
# make modules
```

Esta também é uma longa etapa, deu tempo de ver uns três episódios de The Big Bang Theory. Agora o sétimo passo é instalar os módulos no kernel novo:

```
# make modules_install
```

Em seguida, vamos instalar o kernel:

```
# make install
```

Como oitavo passo, vamos acessar o diretório `/boot` e gerar a imagem inicial do kernel `intrd` (de *initial ram disk*).

```
# mkinitramfs -o initrd.img-4.12.7 4.12.7
```

E, para finalizar, execute o comando:

```
# update-grub
```

Este comando criará a opção de inicialização do seu novo kernel. Após reiniciar o computador, o novo kernel já vai iniciar, basta fazer a sua escolha no menu de inicialização. Vale lembrar que uma nova versão do kernel não necessariamente trará recursos em nível de usuário. Você terá normalmente atualizações de segurança, correções de bugs e melhor desempenho. Sempre é importante ler no site oficial as novidades de cada versão de kernel.

13.3 CONCLUSÃO

Kernel e módulos do kernel são parte do exame LPIC1 e *comptia linux+*, então compreender os comandos aqui descritos no livro é de grande importância para um bom resultado na prova. Além disso, saber compilar um kernel pode ajudá-lo a resolver problemas de incompatibilidade com novos hardwares, como impressoras, por exemplo, já que a cada nova versão mais e mais dispositivos são suportados.

Capítulo 14

HARDENING

Hardening é um termo que se aplica ao processo de busca de ameaças e execução de tarefas corretivas para o servidor, de modo a evitar ataques externos. Normalmente, este processo inclui remover ou desabilitar contas de usuários que não estejam mais ativas e serviços desnecessários.

Configurando o `sudo`

O comando `sudo` permite a usuários comuns obter privilégios de usuário administrador durante a execução de uma determinada tarefa. O nome é uma forma abreviada de se referir a **substitute user** ou **super user**. Observações importantes sobre o comando `sudo`:

- A senha é armazenada por 15 minutos. Após isto, você terá que digitá-la novamente;
- Sua senha não será mostrada quando você a digita no terminal;
- Para executar o `sudo` no gestor de janelas GNOME use `gksudo`;
- Para executar o `sudo` no gestor de janelas KDE, use o comando `kdesudo`.

Vamos instalar o comando `sudo` com o comando:

```
# apt-get install sudo
```

O arquivo de configuração do `sudo` é o `/etc/sudoers`. Abra este arquivo com seu editor preferido ou execute o comando:

```
# visudo
```

Observe no arquivo a linha:

```
root ALL=(ALL) ALL
```

Esta linha define que o usuário `root` tem todas as permissões ao executar o comando `sudo`. Você pode adicionar seu usuário comum para ter poderes de `root` temporariamente ao executar o comando `sudo`, para isto, adicione abaixo desta linha:

```
juliano ALL=(ALL) ALL
```

Neste caso, estou atribuindo ao usuário `juliano`, todas as permissões de `root`, quando ele executar o comando `sudo` à frente de um comando. Exemplo:

```
$ sudo apt-get install elinks
```

Nesse exemplo, mesmo logado como `juliano`, que é um usuário comum e não tem permissões para instalar programas, ao colocar `sudo` à frente do comando, temporariamente `juliano` tem poderes de `root`.

Para melhorar a segurança do sistema com o uso do `sudo`, em caso de esquecer o terminal aberto com o usuário `juliano`, podemos configurar o `sudo` para sempre pedir senha ao se digitar um comando. Para isso, adicione ao final do arquivo:

```
Defaults:juliano timestamp_timeout=0
```

Configurando o servidor SSH

Serviço amplamente utilizado pelos administradores de servidores Linux, o SSH é também um alvo para ataques. Uma ótima maneira de garantir maior segurança ao seu servidor é não permitir o acesso como `root` e mudar a porta padrão do SSH. Para isso, abra o arquivo de configuração localizado em: `/etc/ssh/sshd_config` e altere as linhas: `Port` , `LoginGraceTime` e `PermitRootLogin` para as configurações a seguir:

```
Port 53000
LoginGraceTime 10
PermitRootLogin no
```

Respectivamente, alteramos a porta padrão de 22 para 53000, mudamos o tempo máximo para digitar a senha para apenas 15 segundos e negamos o login com o usuário `root` . Para as regras entrarem em vigor, reinicie o serviço do SSH.

Melhorando as senhas

É muito importante criar senhas seguras para nosso servidor Linux. Para ajudar nesta tarefa, instale a aplicação `pwgen` com o comando:

```
# apt-get install pwgen
```

Para criar 4 modelos de senhas com 10 caracteres aleatórios, execute:

```
# pwgen 10 4
```

Verificando portas

Periodicamente você deve procurar portas abertas e programas desnecessários no sistema para desabilitá-los. Para esta tarefa, usaremos o **NMAP**, aquele scanner de portas altamente poderoso. Para sua instalação, execute:

```
# apt-get install nmap
```

Como forma demonstrativa, vou executar uma varredura no sistema à procura de algum serviço desnecessário:

```
# nmap -sV localhost
```

Em uma das linhas de saída do comando, observei o serviço `Exim` , um servidor de e-mail que não estou usando:

```
25/tcp open stmp Exim smtpd 4.72
```

Para desabilitar o serviço da inicialização, executo o comando:

```
# insserv -rv exim4
```

Outro comando importante e conteúdo da **LPIC** e **LTC-1** é o `netstat` . Ele possui alguns parâmetros importantes para verificarmos as conexões estabelecidas em nosso servidor. Execute:

```
# netstat -pultan
```

Observe uma linha da saída desse comando:

```
tcp        0      0 192.168.1.50:58648      199.38.164.157:443
   ESTABELECIDA 2781/spotify
```

Como eu utilizo o serviço de streaming de música Spotify, ele me mostrou qual porta ele está usando: 2781.

Criando log para o comando `su`

Você pode trocar de usuário a qualquer momento no sistema

usando o comando: `su nome-do-usuario` e sua senha. É interessante criar um arquivo de log para verificar os logins realizados e as tentativas que falharam. Assim poderemos observar se algum usuário está tentando obter através de tentativa e erro acesso a um usuário específico.

Abra o arquivo `/etc/login.defs` e descomente a linha:

```
SULOG_FILE /var/log/sulog
```

Após salvar e fechar o arquivo, crie o arquivo `sulog` :

```
# touch /var/log/sulog
```

Agora, basta monitorar este arquivo. Para monitorar em tempo real, use o comando:

```
# tail -f /var/log/sulog
```

Mantenha o sistema atualizado

Sempre que possível, atualize o sistema. Esta é uma boa prática para qualquer administrador de sistemas Linux.

```
# apt-get update
# apt-get upgrade
```

Bloqueando usuários no `cron`

O serviço de agendamento de tarefas do `cron` permite que você especifique quem pode ou não executar agendamentos. Isto é controlado pelos arquivos `/etc/cron.allow` e `/etc/cron.deny` . Para bloquear um usuário usando o `cron` , basta adicionar seu nome de usuário no arquivo `cron.deny` e, para permitir, coloque em `/etc/cron.allow` . Se você deseja desativar todos os usuários do `cron` , adicione a linha `ALL` ao

arquivo `cron.deny`:

```
# echo ALL >> /etc/cron.deny
```

Desabilitar a detecção de dispositivos USB

Para evitar roubo de informação, você pode desejar desabilitar os usuários de usarem dispositivos USB. Para isso, crie o arquivo: `/etc/modprobe.d/no-usb` e adicione a linha a seguir:

```
install usb-storage /bin/true
```

Remover a interface gráfica

Em um servidor, não é necessária a execução de interface gráfica como o KDE ou GNOME. Para iniciar o sistema sempre pelo modo texto, execute:

```
# systemctl set-default multi-user.target
```

Desligue o IPV6

Se você não estiver usando o IPV6, então você deve desativá-lo. Abra o arquivo `/etc/sysctl.conf` e coloque as linhas:

```
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
net.ipv6.conf.eth0.disable_ipv6 = 1
```

Salve o arquivo e execute o comando:

```
# sysctl -p
```

Bloquear uma conta

O recurso de bloquear uma conta de usuário é muito

importante. Em vez de você remover a conta do sistema, você pode bloqueá-la. Para isso, use:

```
# passwd -l nome_da_conta
```

Lembre-se de que, mesmo bloqueando a conta, o usuário `root` poderá acessá-la normalmente. Para ativar a conta, use:

```
# passwd -u nome_da_conta
```

Tempo para trocar a senha

Você pode definir de quanto em quanto tempo seu usuário deverá trocar sua senha. Usamos o comando `chage` para estas alterações. Para verificar um usuário, execute:

```
# chage -l nome-do-usuario
```

Para configurar o tempo mínimo de troca de senha em 3 dias, por exemplo, execute:

```
# chage -m 3 nome-do-usuario
```

Configurar a data de expiração de senha para uma data específica:

```
# chage -E 2017/12/31 nome-do-usuario
```

Verificar contas sem senha

Uma conta sem senha pode ser uma porta aberta para um acesso não autorizado. Faça a verificação com o comando:

```
# cat /etc/shadow | awk -F: '($2==""){print $1}'
```

Aviso de segurança para usuário remoto

É uma boa opção colocar aviso de segurança para quem desejar fazer um acesso remoto. Estes avisos podem ser usados em algum processo legal, contra um invasor. Coloque seu aviso no arquivo `/etc/issue.net`, em seguida, edite o arquivo `/etc/ssh/sshd_config` colocando a linha:

```
Banner /etc/issue.net
```

Reinicie o serviço do SSH.

Ignore ICMP ou Broadcast Request

ICMP, sigla para o inglês *Internet Control Message Protocol*, é um protocolo integrante do Protocolo IP, definido pelo RFC 792. É utilizado para fornecer relatórios de erros à fonte original. O Broadcast Request é usado para você obter informações sobre todos os hosts da rede. Ao ignorar estas informações, sua conexão é otimizada.

Adicione a seguinte linha no arquivo `/etc/sysctl.conf` para ignorar a solicitação de *ping* ou *broadcast request*:

```
net.ipv4.icmp_echo_ignore_all = 1
net.ipv4.icmp_echo_ignore_broadcasts = 1
```

Salve e execute o comando:

```
# sysctl -p
```

14.1 CONCLUSÃO

Todos os dias, novas técnicas de ataque e invasão são descobertas. Fique atento em canais de segurança da informação na Web e, acima de tudo, mantenha seu sistema atualizado. O site

linux.com pode ser um bom ponto de partida, já que estão sempre antenados na questão de segurança para servidores Linux.

Capítulo 15

SERVIDOR NFS - COMPARTILHANDO ARQUIVOS

NFS (acrônimo para *Network File System*) é um sistema de arquivos distribuídos desenvolvido inicialmente pela Sun Microsystems, INC., a fim de compartilhar arquivos e diretórios entre computadores conectados em rede, formando um diretório virtual. Para que você possa usar o compartilhamento NFS, são necessárias uma máquina cliente e outra servidor.

Configurando o servidor

O primeiro passo é criar os diretórios a serem compartilhados na máquina servidor:

```
# mkdir -p /storage/juliano
```

Agora partiremos para a instalação do servidor NFS e suas configurações.

```
# apt-get install nfs-kernel-server
```

Com o servidor instalado, vamos configurar o compartilhamento do diretório criado no arquivo `/etc/exports` ,

inserindo o IP ao qual vamos dar permissão de acesso. Exemplo:

```
/storage/juliano 192.168.1.2(rw,root_squash,no_subtree_check,asyn
c)
```

As opções mais usadas são:

Opção	Definição
`ro`	Compartilhar apenas leitura.
`rw`	Compartilhar para leitura e gravação.
`root_squash`	Opção padrão, faz com que o usuário `root` não tenha privilégios dentro do compartilhamento NFS montado.
`no_root_squash`	Permite ao usuário `root` os mesmos privilégios que em um diretório local.
`async`	Útil em redes locais pois permite que o NFS transfira arquivos de forma assíncrona, sem precisar esperar pela resposta do cliente a cada pacote enviado. Aumenta um pouco a velocidade da transferência de dados.

Agora, vamos executar o comando para reler o arquivo `/etc/exports` e aplicar o compartilhamento:

```
# exportfs -r
```

Pronto! Diretório compartilhado. Para realizar uma verificação, execute:

```
# showmount -e localhost
Export list for localhost:
/storage/juliano 192.168.1.2
```

Configurando o cliente

Instale o serviço de cliente NFS:

```
# apt-get install nfs-common
```

Agora, vamos criar um ponto de montagem:

```
# mkdir /mnt/storage
```

Verifique o compartilhamento colocando o IP do servidor:

```
# showmount -e 192.168.1.1
Export list for localhost:
/storage/juliano 192.168.1.2
```

Monte o compartilhamento:

```
# mount -t nfs 192.168.1.1:/storage/juliano /mnt/storage
```

Você já pode acessar o compartilhamento no seu ponto de montagem, mas perceberá que o usuário `root` não tem permissão de escrita. Para que seja possível, no servidor, edite novamente o arquivo `/etc/exports`, adicionando a opção `no_root_squash`:

```
/storage/juliano 192.168.1.2(rw,no_root_squash,no_subtree_check,a
sync)
```

Em seguida, execute o comando:

```
# exportfs -r
```

Agora será possível criar conteúdo com o usuário `root` no diretório compartilhado. Para que o compartilhamento seja montado no *boot*, vamos adicionar a configuração no final do arquivo `/etc/fstab`:

```
192.168.1.1:/storage/juliano /mnt/storage nfs defaults,soft 0 0
```

15.1 CONCLUSÃO

O servidor NFS é simples e prático para quem deseja compartilhar arquivos e diretórios em máquinas Linux. Em muitos casos, o NFS é uma opção mais rápida e prática ao servidor de

compartilhamento de arquivos SAMBA, que ainda abordaremos neste livro.

Capítulo 16

SERVIDOR RAID

Do inglês *Redundant Array of Independent Disks*, RAID pode ser traduzido como **Conjunto Redundante de Discos Independentes**. A ideia básica por trás do RAID é combinar diversos discos e de baixo custo em conjunto, a fim de atingir objetivos de desempenho inatingíveis com um disco grande e de custo alto. Este conjunto de discos aparece para os computadores como uma única unidade ou disco de armazenamento lógico.

As principais vantagens do RAID são:

- Ganho de desempenho no acesso para leitura ou gravação;
- Redundância em caso de falha em um dos discos;
- Uso múltiplo de várias unidades de discos;
- Facilidade em recuperação de conteúdo perdido;
- Impacto reduzido na falha de um disco.

Temos duas formas de se montar um servidor RAID, via software e via hardware. Adotaremos o modo de software com o Linux gerenciando os discos. Neste modelo, o RAID entra em funcionamento depois que o kernel é carregado na memória do computador. A principal vantagem é a facilidade de configuração e a flexibilidade, já que podemos trabalhar com vários discos diferentes. A principal desvantagem é a dependência da correta

configuração do sistema operacional.

No RAID via hardware, é necessária uma placa controladora que conecta um disco ao outro. A principal vantagem é o desempenho, já que é mais rápido e independente do sistema operacional. A principal desvantagem é que a placa controladora se torna um **SPOF** (*Single Point of Failure*), ou seja, é necessário ter uma controladora de discos igual ou compatível com a que você possui para o caso de falhas neste hardware.

Tipos de RAID

Os principais níveis de RAID utilizados hoje no mercado são os níveis 0, 1, 5, 6 e suas derivações, como o RAID 10.

RAID 0 (*Stripping*): este é o único nível de RAID que não implementa redundância. Sua finalidade é aumentar o desempenho de leitura e gravação, uma vez que, ao gravar, divide os dados em partes iguais e armazena cada fragmento em um disco diferente, simultaneamente. Por isso, com dois discos, a velocidade de leitura praticamente dobra. Com três discos, triplica, e assim por diante. São necessários ao menos dois discos para implementar RAID 0 e eles podem ser de tamanhos diferentes.

Observação: RAID 0 não garante redundância.

RAID 1 (*Mirroring*): o nível mais utilizado. Sua principal finalidade é prover redundância de dados. Esta é garantida pela duplicação dos dados que são gravados em cada par de discos, logo, se um deles falhar, o outro continuará operando e mantendo a informação disponível, até que a substituição do disco defeituoso seja feita. O ganho de desempenho está na leitura, uma vez que os

dados são lidos em partes iguais e simultaneamente de todos os discos. A desvantagem desse nível é que só metade do volume total de armazenamento nos discos utilizados ficará disponível para o sistema operacional. É preciso no mínimo dois discos para implementar RAID 1, sempre em pares.

Observação: RAID 1 não é backup, é apenas redundância. Backup é você manter uma cópia de seus dados para uma futura restauração; redundância é garantir segurança durante o armazenamento dos dados.

RAID 5: neste nível de RAID, teremos um balanço das vantagens dos níveis anteriores, ou seja, RAID 5 provê um ganho de desempenho e tolerância a falhas a custos menores que RAID 0 ou RAID 1 individualmente. O ganho de desempenho está mais uma vez na leitura. Quanto mais discos forem adicionados à composição, mais rápida será a leitura, uma vez que a gravação é distribuída em blocos de tamanhos iguais por todos os discos.

RAID 6: é um padrão relativamente novo, suportado apenas por algumas controladoras. Ele é semelhante ao RAID 5, porém usa o dobro de bits de paridade, garantindo a integridade dos dados caso até 2 dos HDs falhem ao mesmo tempo. Tanto no RAID 5 quanto no RAID 6, o servidor continua funcionando normalmente durante todo o processo de substituição de disco.

RAID 10: combina as vantagens do RAID 0 e RAID 1 em um único sistema. Fornece segurança efetuando espelhamento de todos os dados em um conjunto secundário de discos enquanto utiliza listagem em cada conjunto de discos para acelerar as transferências de dados.

16.1 CRIANDO NOSSO LABORATÓRIO

Para conhecermos de forma prática o uso do RAID, é possível utilizarmos um ambiente de virtualização. No Virtualbox, para este experimento, criei 10 discos. Após isto, criei 1 partição para cada disco, usando o `cfdisk` . Exemplo:

```
# cfdisk /dev/sdb
# cfdisk /dev/sdc
# cfdisk /dev/sdd
```

Realizei o processo com os meus 10 discos criados até `/dev/sdk` . Com todos os discos criados e particionados, instalei o utilitário `mdadm` , que será responsável por gerenciar todos os discos:

```
# apt-get install mdadm
```

Durante o processo de instalação, aceite o padrão (`ALL`) e conclua a instalação. Você poderá acompanhar em tempo real os discos que estão no RAID, pelo arquivo `/proc/mdstat` .

Criando um RAID nível 0

Vamos iniciar nosso estudo criando um RAID de nível 0. Para isso, executaremos no terminal o comando:

```
# mdadm --create /dev/md0 --level=0 --raid-devices=2 /dev/sdb1 /d
ev/sdc1
```

Nesse comando, estamos criando o RAID de nível 0, com o nome `/dev/md0` e que usará os discos `/dev/sdb1` e `/dev/sdc1` . Após a execução do comando, você pode observar que eles estão ativos no RAID, visualizando o arquivo `/proc/msdstat` . Exemplo:

```
# cat /proc/mdstat
```

Criando um RAID nível 1

Basicamente, o comando de criação é o mesmo para todos os níveis de RAID, logo, o comando para criar um RAID de nível 1 é:

```
# mdadm --create /dev/md1 --level=1 --raid-devices=2 /dev/sdd1 /d
ev/sde1
```

Observe que alteramos a opção `level` para 1, assim como colocamos outro nome (`md1`) e alteramos os discos utilizados: `/dev/sdd1` e `/dev/sde1` .

Você pode verificar a sua criação:

```
# cat /proc/mdstat
```

Criando um RAID de nível 5

O comando segue a mesma estrutura dos anteriores:

```
# mdadm --create /dev/md2 --level=5 --raid-devices=3 --spare-devi
ces=1 /dev/sdh1 /dev/sdi1 /dev/sdj1 /dev/sdk1
```

Observe que temos uma opção a mais e usamos agora 4 discos. A opção `spare-devices` é usada para definir os discos reservas, ou seja, que serão utilizados apenas quando um dos discos de uso falhar. Atribuímos 1 disco para `spare-devices` , ele sempre usará o último da atribuição do comando, neste caso: `/dev/sdk1` .

Vendo os detalhes do RAID

Para ver mais detalhes sobre o RAID criado, utilize o comando:

```
# mdadm --detail /dev/md3
```

No meu caso, observo na saída do comando que tenho 3 discos ativos e 1 *spare* (reserva).

Formatando o RAID

Agora que os RAIDS estão criados, vamos aplicar um sistema de arquivos:

```
# mkfs.ext4 /dev/md1
```

Use o mesmo comando para os discos `/dev/md2` e `/dev/md3` .

Criando ponto de montagem

O RAID funciona de forma semelhante a uma partição comum. Portanto, é preciso definir um ponto de montagem para que você possa gravar e acessar arquivos e diretórios. Crie um ponto de montagem e, em seguida, monte o disco:

```
# mkdir /mnt/md1
# mkdir /mnt/md2
# mkdir /mnt/md3
# mount -t ext4 /dev/md1 /mnt/md1
# mount -t ext4 /dev/md2 /mnt/md2
# mount -t ext4 /dev/md3 /mnt/md3
```

Simulando uma falha de disco

Para aprendermos a trocar um disco em um servidor RAID sem desligar o sistema, vamos simular uma falha de disco em nosso `/dev/md3` , que usa RAID nível 5 e tem disco reserva.

```
# mdadm /dev/md3 --fail /dev/sdh1
```

Nesse exemplo, estamos encaminhando uma falha ao disco `/dev/sdh1`, que para de funcionar imediatamente. Se você executar agora:

```
# mdadm --detail /dev/md3
```

Perceberá que o disco reserva `/dev/sdk1` assumiu o lugar do `/dev/sdh1`, que agora se encontra sem uso.

Substituir disco com falha

Para remover o disco com falha, usaremos o comando:

```
# mdadm /dev/md3 --remove /dev/sdh1
```

Em seguida, você pode adicionar um novo disco ao RAID, que assumirá posição de reserva, com o comando:

```
# mdadm /dev/md3 --add /dev/sdh1
```

Nesse exemplo, usei o mesmo disco `/dev/sdh1`, mas em uma situação real, você utilizará o caminho relativo ao seu disco adicionado.

Criando um RAID de nível 10

O RAID de nível 10 necessita de 2 RAIDS de nível 1 existentes, e a aplicação sobre eles de um RAID de nível 0. Para isso, vamos criar mais um RAID 1 com os discos `/dev/sdf1` e `/dev/sdg1`:

```
mdadm --create /dev/md4 --level=1 --raid-devices=2 /dev/sdf1 /dev
/sdg1
```

Agora que temos 2 discos de nível 1 (`/dev/md1` e `/dev/md4`) vamos aplicar sobre eles o RAID 0:

```
mdadm --create /dev/md5 --level=0 --raid-devices=2 /dev/md1 /dev/
md4
```

Pronto. Seu RAID 10 foi criado.

Arquivando tudo

Para que você possa tornar esta configuração permanente, edite o arquivo `/etc/mdadm/mdadm.conf`, colocando a configuração:

```
DEVICE/dev/sdb1    /dev/sdc1
ARRAY/dev/md0    devices=/dev/sdb1,/dev/sdc1

DEVICE /dev/sdd1/dev/sde1
ARRAY /dev/md1    devices=/dev/sd1 /dev/sde1

DEVICE /dev/sdh1 /dev/sdi1 /dev/sdhj1 /dev/sdk1
ARRAY /dev/md3    devices=/dev/sdh1, /dev/sdi1, /dev/sdj1, /dev/s
dk1

DEVICE /dev/sdf1 /dev/sdg1
ARRAY /dev/md4    devices=/dev/sdf1 /dev/sdg1
```

16.2 CONCLUSÃO

RAID não é backup, é redundância! Isso significa que, usando RAID, você terá maior velocidade e segurança durante o processo de cópia de arquivos. Observe o caso do RAID 5: mesmo quando um disco falha, o sistema fica operando normalmente, até que ele seja substituído. As certificações de Linux exigem do candidato um conhecimento básico sobre RAID, por isso, tenha atenção aos comandos do `mdadm` (aplicação que gerencia os discos RAID) aqui abordados.

Capítulo 17

LOGICAL VOLUME MANAGER - LVM3

Logical Volume Manager (LVM), em português Gerenciador de Volume Lógico, é um grande disco virtual, que pode ter mais de um dispositivo de armazenamento. A vantagem de se utilizar LVM é a facilidade em se redimensionar este volume de dados, adicionando ou removendo discos. Com LVM, empresas diminuem custos. Por exemplo: você está implementando um servidor de arquivos e não sabe ao certo quanto de espaço será usado por seus usuários. Em vez de comprar um disco com uma grande capacidade de armazenamento, e caro, você pode comprar um disco menor e ir expandindo espaço conforme a sua real necessidade.

Usando o LVM2

Para criar um laboratório de teste, vamos criar 3 discos no Virtualbox e inicializar o sistema. Agora vamos criar as partições nestes discos. Não é necessário aplicar um sistema de arquivos.

```
# cfdisk /dev/sdb
# cfdisk /dev/sdc
# cfdisk /dev/sde
```

Tendo as respectivas partições, `/dev/sdb1`, `/dev/sdc1` e `/dev/sde1`, vamos instalar o LVM2:

```
# apt-get install lvm2
```

Após a instalação do LVM2, adicione as suas partições ao LVM com o comando `pvcreate`:

```
# pvcreate /dev/sdb1
# pvcreate /dev/sdc1
# pvcreate /dev/sde1
```

Observe que o nome do comando é `pvcreate`, de *Physical Volume*. Você pode ver os discos adicionados ao LVM usando o comando:

```
# pvdisplay
```

Agora que temos 3 partições que serão usadas pelo LVM, vamos usá-las para a criação de uma grande massa de dados:

```
# vgcreate storage /dev/sdb1 /dev/sdc1
```

Observe que o nome do comando `vgcreate` significa *Volume Group*, ou seja, criamos um grande volume usando as duas partições com o nome `storage`. Não coloquei a partição `/dev/sde1` intencionalmente, para demonstrar daqui a pouco como podemos adicionar uma partição a um volume já existente. Para verificar este Volume Group, use o comando:

```
# vgdisplay -v storage
```

Com ele criado, vamos definir o tamanho de nosso **volume lógico**. Ou seja, o espaço que usaremos para armazenar nossas informações.

```
# lvcreate -L 3G -n lv_storage storage
```

Nesse comando, estamos criando um *Logical Volume*, LV, de 3G chamado `lv_storage` que está pegando espaço do Volume Group `storage`. Para visualizar o volume lógico, use:

```
# lvdisplay -v /dev/storage/lv_storage
```

Você também pode visualizar no Volume Group que estamos usando 3G de espaço:

```
# vgdisplay -v storage
```

Agora, podemos aplicar um sistema de arquivos e começar a usar este espaço lógico:

```
# mkfs.ext4 /dev/storage/lv_storage
```

Crie um ponto de montagem e acesse o espaço como uma partição comum:

```
# mkdir /mnt/lv_storage
# mount -t ext4 /dev/storage/lv_storage /mnt/storage
```

Digamos que você chegou a 90% de uso deste espaço e precisa urgentemente de mais espaço para você ou seus usuários. Neste caso, basta adicionar mais discos ao LVM. Vamos usar aquela partição `/dev/sde1`. Para isso, faça o comando:

```
# vgextend storage /dev/sde1
```

Agora que adicionamos a partição `/dev/sde1` você pode visualizar com:

```
# vgdisplay -v storage
```

Para adicionar mais espaço ao volume lógico que criamos, o primeiro passo é desmontar o acesso:

```
# umount /mnt/lv_storage
```

Com o dispositivo desmontado, execute o comando:

```
# lvextend -L +2G /dev/storage/lv_storage
```

Com esse comando, estamos adicionando 2G ao nosso Volume Group. Para que o kernel do Linux identifique este novo espaço, sem a necessidade de reiniciar o computador, digite:

```
# e2fsck -f /dev/storage/lv_storage
# resize2fs /dev/storage/lv_storage
```

Agora, basta você montar o volume:

```
# mount -t ext4 /dev/storage/lv_storage /mnt/storage
```

17.1 CONCLUSÃO

Em algum momento podemos nos encontrar em uma situação emergencial por conta do crescimento excessivo de arquivos em decorrência da produtividade do servidor, e o cálculo que fizemos no ato da instalação pode ser insuficiente. Para estarmos tranquilos em situações como esta, o melhor recurso é o LVM2, que poucos administradores Linux utilizam, mas deveria ser um pré-requisito para a instalação de qualquer servidor Linux, já que facilita o gerenciamento dos discos a qualquer momento.

Capítulo 18

SERVIDOR SAMBA - CONTROLADOR DE DOMÍNIO

O servidor SAMBA é utilizado em sistemas operacionais do tipo Unix como Linux e BSD, simulando um servidor Windows e permitindo que seja feito o gerenciamento e compartilhamento de arquivos em uma rede mista. Assim, podemos ter computadores Windows e Linux trocando dados e compartilhando impressoras, por exemplo. A versão atual é o SAMBA4 que não só provê arquivos e serviços de impressão para vários clientes Windows, mas também pode integrar-se com o Windows Server Domain, tanto como *Primary Domain Controller* (PDC) ou como um Domain Member. Pode fazer parte também de um *Active Directory Domain*.

O SAMBA foi criado em cima do protocolo SMB (*Server Message Block*) através de engenharia reversa. Este protocolo era utilizado para a troca de arquivos, impressoras e portas seriais em computadores executando o Microsoft Windows. Seu criador Andrew Tridgell escolheu a palavra SAMBA porque tinha fácil pronúncia e também é uma clara referência ao protocolo SMB.

Pré-requisitos

Vamos criar um controlador de domínio com o SAMBA4 usando a o Debian 8.9 - Versão estável. A versão 9 ainda possui alguns bugs com o SAMBA4 que provavelmente serão corrigidos em breve; por este motivo, recomendo a versão estável 8.9:

https://cdimage.debian.org/mirror/cdimage/archive/8.9.0/amd64/iso-cd/debian-8.9.0-amd64-CD-1.iso

Após realizar a instalação do Debian, vamos iniciar o processo de instalação do servidor SAMBA, usando os comandos:

```
# apt-get install samba smbclient krb5-user dnsutils winbind
```

Durante o processo de instalação dos pacotes, o Kerberos (software necessário para o Active Directory) vai solicitar um nome de domínio. Vamos colocar como exemplo:

```
resident.intra
```

O Kerberos vai solicitar o nome da máquina e o nome da máquina administrativa. Em ambos os casos, coloque `localhost`. Caso você tenha digitado algo errado e o instalador se fechar de modo inesperado, abra-o novamente, executando o comando:

```
# dpkg-reconfigure krb5-config
```

Backup do `smb.conf`

De volta à linha de comando, vamos criar um backup do arquivo de configuração padrão do SAMBA. Caso aconteça algum problema inesperado, podemos restituir o arquivo original.

```
# cp /etc/samba/smb.conf /etc/samba/smb.conf.bkp
```

Agora, removeremos o arquivo de configuração padrão:

```
# rm /etc/samba/smb.conf
```

O samba-tool

O próximo passo para a criação do nosso controlador de domínio é executarmos a ferramenta `samba-tool` , que auxilia na criação do arquivo de configuração do SAMBA, mas de modo interativo. Utilize a ferramenta com o parâmetro:

```
# samba-tool domain provision
```

Ao solicitar o nome de domínio, coloque o mesmo que usou no Kerberos:

```
resident.intra
```

Aceite a opção padrão para definir o servidor como um *DC - Domain Controller*. A próxima opção é a escolha do DNS, sendo a melhor opção usarmos o padrão, que é o DNS interno do próprio SAMBA. Após esta escolha, a ferramenta `samba-tool` vai oferecer uma opção de DNS externo (Web); informe o seu DNS atual ou outro como o do Google, por exemplo (8.8.8.8).

O `samba-tool` vai solicitar uma senha para o controlador de domínio. Esta parte necessita de uma atenção especial, já que a senha deve possuir pelo menos seis caracteres, com números e letras. Caso sua senha não seja adequada, o `samba-tool` falhará, e você terá que executá-lo novamente.

Reinicie o seu servidor:

```
# reboot
```

Testando a conexão

Após o servidor reiniciar, podemos realizar um pequeno teste usando o comando:

```
# smbclient -L localhost -U Administrator
```

Se tudo estiver correto, será solicitada a senha que você colocou no utilitário `samba-tool` e você terá uma informação sobre o seu domínio.

Configurando o DNS local

Como definimos que o SAMBA vai controlar também o nosso DNS, é importante editarmos o arquivo de configuração `/etc/resolv.conf` e apontarmos para nossa própria máquina, ou seja, `localhost` :

```
nameserver 127.0.0.1
```

Outra alteração importante de ser realizada é a edição do arquivo `/usr/share/samba/setup/krb5.conf` . Dentro dele, altere a variável para o endereço exato do domínio. Por exemplo:

```
[libdefaults]
    default_realm = RESIDENT.INTRA
```

Observe que coloquei o domínio em letra maiúscula, do mesmo modo que está no arquivo de configuração: `/etc/samba/smb.conf` . Vamos copiar este arquivo para o diretório `/etc` :

```
# cp /usr/share/samba/setup/krb5.conf /etc
```

Colocar uma máquina no domínio

Configure sua máquina Microsoft Windows com o DNS do servidor SAMBA, no meu caso: 192.168.1.60. Lembre-se de que a máquina deve estar na mesma rede local e com IP dentro do mesmo *range*. Após fazer a alteração do DNS, configure-a para o domínio: `RESIDENT.INTRA` ou o nome que você escolheu.

Neste momento, o Windows solicitará um usuário e senha, coloque: `Administrator` e sua senha.

Criando um compartilhamento

De volta ao servidor SAMBA, vamos criar um compartilhamento através do arquivo de configuração `/etc/samba/smb.conf` . Para um compartilhamento público, apenas adicione as linhas a seguir ao final do seu arquivo:

```
[publico]
    path = /storage/samba/public
    read only = No
    browseable = Yes
```

Salve, feche o arquivo e crie o diretório:

```
# mkdir -p /storage/samba/public
```

Agora este compartilhamento já vai estar disponível. Abrindo-o no Microsoft Windows com o usuário `Administrator` , você pode alterar suas permissões.

Criando usuários

Para verificar todos os usuários em seu servidor SAMBA, utilize o comando:

```
# samba-tool user list
```

Para adicionar um usuário ao servidor, utilize o comando:

```
# samba-tool user add barry
```

Criação de grupos

Para criarmos grupos usando o SAMBA, podemos usar o comando:

```
# samba-tool group add financeiro
```

Neste exemplo, estamos criando um grupo chamado `financeiro` . Para visualizar os membros deste grupo, podemos usar o comando:

```
# samba-tool group listmembers financeiro
```

Para adicionar membros a este grupo, usamos o comando:

```
# samba-tool group addmembers financeiro barry
```

Compartilhamento para o grupo

O primeiro passo é editar o arquivo `/etc/samba/smb.conf` criando um compartilhamento:

```
[financeiro]
path = /storage/samba/financeiro
read only = No
browseable = Yes
```

Agora no Windows, clique com o botão direito sobre o diretório `financeiro` , depois siga em `Propriedades - Segurança - Editar - Adicionar` , pesquise o grupo e adicione as permissões necessárias.

18.1 CONCLUSÃO

O servidor SAMBA permite o gerenciamento e compartilhamento de recursos em redes formadas por computadores Windows, sendo um substituto muitas vezes até melhor que o próprio Windows Server. Isso se deve à sua segurança e também pelo custo quase zero de sua implementação. Neste capítulo, foram abordados apenas alguns dos recursos desta poderosa ferramenta, que são os mais exigidos nas empresas, mas vale muito a pena se dedicar um pouco à documentação oficial, que está disponível em: https://www.samba.org/samba/docs/.

CAPÍTULO 19

SERVIDOR APACHE

Criado em 1995 por Rob McCool, na época funcionário da NCSA (*National Center for Supercomputing Applications*), o servidor Apache ou Servidor HTTP Apache é o mais bem-sucedido servidor Web livre que existe. Trata-se de um servidor Web muito popular, utilizado principalmente no Linux.

Assim como qualquer servidor do tipo, o Apache é responsável por disponibilizar páginas e todos os recursos que podem ser acessados pelo internauta. Envio de e-mails, mensagens, compras online e diversas outras funções podem ser executadas graças a servidores como o Apache.

O que vale destacar no Apache é que, apesar de tudo, ele é distribuído sob a licença GNU, ou seja, é gratuito e pode ser estudado e modificado por meio de seu código-fonte por qualquer pessoa.

19.1 INSTALANDO O SERVIDOR APACHE

A instalação do servidor Apache é bem simples, já que o pacote está disponível nos repositórios Debian e derivados:

```
# apt-get install apache2
```

Após a instalação, abra qualquer máquina de sua rede local e, no navegador, aponte para o IP no qual você acabou de instalá-lo. Você já terá uma página informando a versão do Apache.

Document Root

No servidor `apache2`, as páginas que você criar em `.html` ou outra linguagem devem estar no diretório `/var/www/html`.

Para um pequeno teste, remova a página `index.html` que se encontra neste diretório, crie outra qualquer e em qualquer navegador aponte para o IP de seu navegador Apache.

Apache com suporte a PHP

Páginas em HTML/JavaScript são reconhecidas automaticamente pelo Apache, mas para quem utiliza PHP é necessária a configuração de um módulo adequado. Estes módulos estão disponíveis no repositório Debian e derivados. Neste caso, basta a instalação:

```
# apt-get install php5
```

Dependendo da sua versão do Debian, versões mais recentes do PHP podem estar disponíveis. É importante reiniciar o servidor Apache:

```
# service apache2 restart
```

Para testar se o suporte a PHP está funcionando, acesse o diretório `/var/www/html`, remova a página `index.html` e crie o arquivo `index.php`. Nele, coloque o código a seguir:

```
<?php phpinfo();?>
```

Agora, em qualquer navegador, aponte para o IP do seu servidor Apache. Você deve ter como retorno informações sobre a versão do PHP instalada.

19.2 INSTALAÇÃO DE UM BANCO DE DADOS

Para instalar aplicações de gerenciamento de conteúdo (CMS) como o Wordpress, Moodle, entre tantas outras, no seu servidor Apache, além do PHP, é necessário um banco de dados. Vamos instalar o `mysql-server` .

```
# apt-get install mysql-server
```

Durante o processo de instalação, você vai definir a senha de `root` do `mysql` ; não confundir, com `root` , administrador do sistema.

Após a instalação, você pode abrir o banco de dados, executando o comando:

```
# mysql -u root -p
```

Dentro do prompt do `mysql` , vamos criar um banco de dados de exemplo:

```
create database cliente;
```

Agora que criamos um banco de dados chamado `cliente` , vamos criar um usuário administrador deste banco com todas as permissões:

```
create user 'juliano'@'localhost'IDENTIFIED BY 'jro1984';
```

Observe que `jro1984` é a senha definida para este usuário de nome `juliano` que criamos. Dentro do prompt do `mysql` nunca

se esqueça de usar o ponto e vírgula ao final da linha. Para atribuir todas as permissões ao usuário `juliano` no banco cliente, execute o comando:

```
grant all privileges on cliente.*to 'juliano'@'localhost';
```

Integração PHP-MySQL

Instale a integração entre e o MySQL e o PHP através do comando:

```
# apt-get install php5-mysql
```

Após a instalação, reinicie o servidor Apache:

```
service apache2 restart
```

Instalando o Wordpress

Faça download do CMS Wordpress através do link: https://wordpress.org/latest.tar.gz/.

E descompacte-o dentro do diretório `/var/www/html/` , abra qualquer navegador e aponte para o IP do seu servidor Apache com o subdiretório `wordpress` . Exemplo:

```
192.168.1.89/wordpress
```

O instalador do Wordpress vai iniciar. Coloque as informações que você cadastrou no banco de dados:

- **Nome do banco de dados:** `cliente`
- **Nome do usuário:** `juliano`
- **Senha:** (senha do seu usuário do banco de dados)

- **Servidor de banco de dados:** `localhost`
- **Prefixo das tabelas:** `wp_`

As próximas informações necessárias do instalador do Wordpress são referentes ao título do seu site, nome de usuário para o Wordpress e endereço de e-mail. Após preencher estes campos, seu Wordpress já vai estar instalado e pronto para ser usado.

19.3 CONCLUSÃO

Instalar o servidor Apache e integrá-lo a um banco de dados é uma tarefa simples. Muitas vezes, o maior trabalho é na instalação de um CMS específico, que exige diversas dependências de módulos para o Apache ou PHP. Nestes casos, o mais indicado é ler sempre atenciosamente as documentações disponíveis no site do desenvolvedor.

CAPÍTULO 20

SERVIDOR PROXY

Um *proxy* (em português, "procurador" ou "representante") é um servidor que age como intermediário para requisições de clientes solicitando recursos de outros servidores. Um cliente conecta-se ao servidor proxy, solicitando algum serviço, como uma página Web. O proxy avalia a solicitação através de uma lista de regras previamente criada, e autoriza ou não o cliente a acessar a página.

Mas o servidor proxy pode ser mais do que apenas um filtro de conteúdo. É possível, por exemplo, armazenar nele um cache dos sites mais visitados pelos clientes, de modo a otimizar a navegação e diminuir os custos de banda.

Neste capítulo, vamos utilizar o servidor proxy **Squid**, que sem dúvida é um dos melhores servidores proxy do mercado. Ele foi originalmente escrito para rodar em sistemas operacionais do tipo Unix como o Linux, mas desde sua versão 2.6 funciona também em sistemas Windows.

Instalação do Squid

O processo de instalação do Squid no Debian e derivados é bem simples:

```
# apt-get install squid3
```

O arquivo de configuração padrão fica localizado em:

```
/etc/squid3/squid.conf
```

Quando você abrir este arquivo pela primeira vez, pode se assustar com o tamanho. São mais de 3200 linhas! Mas não se preocupe. Praticamente todas as linhas são de comentários que demonstram as funcionalidades do Squid.

O Squid trabalha com ACLs (*Access Control List*) que são as regras para a navegação via proxy. Você poderá criar quantas desejar. Estas regras são definições de sites ou até mesmo de palavras na URL que o usuário poderá acessar ou não.

As ACLs são dispostas da seguinte forma:

```
acl NOME_DA_ACL TIPO_DA_ACL parâmetro
```

Ou

```
acl NOME_DA_ACL TIPO_DA_ACL "/caminho/completo/regra.conf"
```

Na primeira regra, serão definidos todos os parâmetros em sequência, separados por espaço. Esta forma é utilizada para regras com poucos parâmetros. Na segunda regra, será definido um arquivo para adição dos parâmetros dispostos um por linha, logo esta forma serve para regras com muitos parâmetros.

Todas as ACLs são tratadas com *case sensitive*. Para definir como *case insensitive* utilize a opção `-i` logo após o tipo de ACL.

Na linha 1040 do arquivo você terá a ACL:

```
#acl localnet src 192.168.0.0/16
```

Observe que esta ACL está comentada com o símbolo de sustenido à frente (#), o que significa que ela não está funcionando. Vamos editá-la colocando a configuração de nossa rede.

```
acl localnet src 172.16.10.0/24
```

Lembre-se de configurar com o IP de sua rede. Outro passo importante é liberar as regras para `localnet` . Na linha 1209, encontramos:

```
#http_access allow localnet
```

Descomente-a:

```
http_access allow localnet
```

Agora é preciso liberar e definir o tamanho do cache, ou seja, quanto de espaço o Squid vai armazenar de sites visualizados pelos clientes. Na linha 3061, temos:

```
# maximum_object_size 4 MB
```

Descomente esta linha e defina um tamanho. Por exemplo:

```
maximum_object_size 800 MB
```

Neste caso, os clientes podem fazer download maiores que 800MB, porém o Squid só fará cache de arquivos até este tamanho. Digamos que um cliente acabou de fazer download de um filme de 700MB de tamanho; quando outro cliente optar por fazer download deste mesmo filme, ele usará o cache do Squid e não o download do servidor remoto (Web), economizando banda e tornando o processo mais rápido.

Na linha 3234 do arquivo de configuração do Squid:

```
# cache_dir ufs /var/spool/squid3 100 16 256
```

Você tem a configuração do tamanho total de armazenamento de dados que o servidor vai permitir. Além do tamanho da árvore de cache em níveis (quanto maior, mais estruturada). Uma boa configuração seria:

```
cache_dir ufs /var/spool/squid3 10000 128 256
```

Neste caso o valor 10000 representa 10GB de tamanho limite ao cache do Squid e os outros valores são os tamanhos de diretórios e subdiretórios dentro da estrutura de cache. Com estas configurações, já é possível colocarmos o nosso servidor para iniciar. Salve as alterações e feche o arquivo.

Parando o serviço e criando o cache

Para criar a estrutura de árvores que será o cache de navegação do Squid, será necessário parar o serviço:

```
# /etc/init.d/squid3 stop
```

Em seguida, execute o comando:

```
# squid3 -z
```

Os diretórios serão criados no caminho `/var/spool/squid3/`. Dê `< enter >` no final do processo. Para iniciar novamente o serviço do Squid, execute:

```
# /etc/init.d/squid3 start
```

Outra maneira de parar e ativar o serviço é com os comandos:

```
# service squid3 stop
# service squid3 start
```

Se você abrir as configurações do navegador Firefox, por exemplo, em qualquer cliente da rede e apontar para o IP em que temos o `squid3` em funcionamento, o proxy já vai estar funcionando e fazendo o cache de navegação. Porém, ainda não terá nenhuma regra de bloqueio.

No meu caso, eu coloquei no navegador do cliente o meu endereço de IP: 172.16.10.1 e porta 3128.

Limpando o arquivo de configuração

Antes de começar a criar regras de navegação, é importante ter um arquivo de configuração mais simples. Crie um backup do arquivo original para evitar qualquer problema futuro:

```
# cp /etc/squid3/squid.conf /etc/squid3/squid.conf.backup
```

Agora, vamos remover todas as linhas de comentários do Squid, o que vai reduzir drasticamente o número de linhas do arquivo. O melhor método para fazer isto é usando o editor Vim:

```
# vim /etc/squid3/squid.conf
```

Com o Vim aberto, no arquivo, tecle `< esc >` e coloque:

```
g/^#/: delete
```

Este comando apagará todas as linhas que iniciam com comentário (`#`). Você vai observar que agora o arquivo possui uma série de linhas em branco (vazias). Para removê-las, tecle `< esc >` e coloque:

```
g/^$/: delete
```

Nosso arquivo de configuração agora deve ter entre 30 e 35 linhas. Sendo assim, muito mais fácil de trabalharmos com ele.

Criando nossas ACLs

O Squid trabalha com as regras de forma linear, ou seja, se na primeira linha você coloca uma regra para bloquear um determinado site, na segunda não pode colocar uma regra liberando-o, pois a segunda linha será ignorada uma vez que na primeira ele já bloqueou o acesso.

Abaixo da linha:

```
acl localnet src 172.16.10.0/24
```

Vamos começar a criar nossa primeira regra. Para exemplo, vamos bloquear a palavra `linux` porém, permitir o acesso ao site `vivaolinux.com.br` . Quando bloqueamos uma palavra, toda vez que houver esta palavra em uma URL, como em `br.linux.org` , `linux.org` , o site será recusado.

```
acl linux url_regex linux
acl liberados dstdomain .vivaolinux.com.br
```

Por enquanto, só criamos a regra. Para que funcione é também necessário criar o bloqueio. Abaixo da linha:

```
http_access allow localnet
```

Crie o bloqueio:

```
http_access allow localnet liberados
http_access deny localnet linux
```

Observe que, na frente de `acl` , colocamos o nome da regra: `linux` , `liberados` . A opção `url_regex` é para o bloqueio de palavras, e `dstdomain` , para o bloqueio ou liberação de domínios. No bloqueio, podemos usar `allow` para liberar e `deny` para negar.

Salve e feche o arquivo de configuração. Para que as novas regras entrem em vigor, não é necessário reiniciar o serviço do Squid, você pode apenas reiniciar a leitura do arquivo de configuração:

```
# squid3 -k parse
# squid3 -k reconfigure
```

Na máquina cliente que tem como proxy o nosso servidor configurado no navegador, observe agora que qualquer site com a palavra `linux` não abrirá, porém, o site `vivaolinux.com.br` funciona normalmente.

Criando listas em arquivos

Um modo prático de organizar nossa configuração do Squid é criar arquivos com listas de sites ou palavras que desejamos bloquear. Deste modo, não precisamos ficar escrevendo no arquivo padrão de configuração do Squid.

No exemplo a seguir, vamos criar 3 arquivos, um para palavras que desejamos bloquear, outro para sites que vamos bloquear e o último para sites cujo acesso vamos liberar.

```
# touch /etc/squid3/palavras_bloqueadas.txt
# touch /etc/squid3/liberados.txt
# touch /etc/squid3/bloqueados.txt
```

Agora que criamos os arquivos, dentro do arquivo de configuração `/etc/squid3/squid.conf` , vamos apontar para as listas:

```
acl palavras_bloqueadas url_regex -i “/etc/squid3/palavras_bloque
adas.txt”
acl liberados dstdomain -i “/etc/squid3/liberados.txt”
acl bloqueados dstdomain -i “/etc/squid3/bloqueados.txt”
```

E claro, criar os bloqueios:

```
http_access allow localnet liberados
http_access deny localnet palavras_bloqueadas
http_access deny localnet bloqueados
```

Salve e reinicie o arquivo de configuração do Squid. Ele pode informar que os arquivos estão vazios durante a reinicialização.

```
# squid3 -k reconfigure
```

Agora, coloque as palavras que você deseja bloquear dentro do arquivo `palavras_bloqueadas.txt` . O mesmo vale para os sites que devem iniciar com ponto.

Exemplo:

```
# vim /etc/squid3/liberados
.certificacoes.net.br

# vim /etc/squid3/bloqueados
.microsoft.com.br

# vim /etc/squid3/palavras_bloqueadas
sexo
microsoft
```

Navegar apenas pelo proxy

Se você possui um servidor *gateway*, é possível bloquear as portas 80 e 443 de modo que os clientes consigam apenas navegar pelo proxy que usa a porta padrão 3128. Neste caso, basta digitar as seguintes regras em seu *firewall*:

```
# iptables -A FORWARD -p tcp --dport 80 -j DROP
# iptables -A FORWARD -p tcp --dport 443 -j DROP
```

Para não perder esta regra no *boot* do sistema, você pode adicioná-las ao arquivo: `/etc/rc.local` .

20.1 CONCLUSÃO

Praticamente todas as instituições possuem regras de acesso a sites, como redes sociais e sites com conteúdo pornográfico. Apesar das famosas reuniões de produtividade em que estas regras são sempre lembradas, manter um proxy é sempre uma solução eficaz aos usuários que sempre procuram um modo de burlar as regras. O Squid 3, assim como todos os servidores que abordamos neste livro, é de código-fonte aberto e seu custo de implementação é zero. Como em todos os outros servidores que abordei neste livro, assumi o ponto de vista de um usuário iniciante, por este motivo o conteúdo é básico e prático.

Espero que o conteúdo de todo este livro, que foi pensado e escrito de uma forma linear, ou seja, iniciando pelos primeiros comandos e finalizando com servidores, tenha sido satisfatório. O trabalho de um administrador de redes Linux é um constante aprendizado, todos os dias. São novas atualizações e novas ferramentas, por isto é indispensável ter disciplina para estudar e documentar todos os seus passos de aprendizado.

Até o próximo livro, prof. Juliano Ramos - Vamos que vamos!